Universität Münster

UNIVERSITY OF TWENTE.

Josef Al Thaher

Exploring Retrieval-Augmented Generation for Large Language Models: Enhancing University IT Support with a RAG-based Chatbot

Master Thesis

at the Department of Information Systems
(University of Münster)

Principal Supervisor:	Prof. Dr. rer. nat. Raimund Vogl
Associate Supervisor:	Prof. Dr. ir. M. van Keulen
Tutor:	Jonathan Radas, M.Sc.
Presented by:	Josef Al Thaher [MS:430410, UT:3268225] Weseler Straße 569 48163 Münster +49 1575 2093217 j_alth12@uni-muenster.de j.althaher@student.utwente.nl
Submission:	22nd April 2025

A computer would deserve to be called intelligent if it could deceive a human into believing that it was human.

Alan Turing

Contents

Figures

Tables

Listings

Abbreviations

AGI Artificial General Intelligence

AI Artificial Intelligence

ANN Approximate Nearest Neighbour

CIT Center for Information Technology

CNN Convolutional Neural Network

DSR Design Science Research

GPT Generative Pre-trained

HCI Human-Computer Interaction

LCEL LangChain Expression Language

LLM Large Language Model

LSTM Long Short-Term Memory

NLP Natural Language Processing

NLU Natural Language Understanding

RAG Retrieval-Augmented Generation

RNN Recurrent Neural Network

1 Introduction

The advent of Large Language Models (LLMs) has revolutionized the field of Natural Language Processing (NLP) (Hadi et al., 2023). These models utilize deep learning methods, especially transformer architectures, to capture and interpret the complex patterns and structures inherent in language data (Dong et al., 2023; Luitse & Denkena, 2021). LLMs can process large volumes of unstructured text and multimodal data, capturing semantic relationships and greatly improving machine understanding and generation of human-like language (Hadi et al., 2023). However, a fundamental challenge persists: the trade-off between generality and specificity. LLMs excel at answering general questions but often falter when faced with domain-specific inquiries or knowledge-intensive tasks for which they lack explicit knowledge (Gu et al., 2021). This gap in performance is particularly pronounced in scenarios where up-to-date domain-specific information is relevant (X. Wang et al., 2024). One common approach to addressing this challenge is transfer learning (Ruder et al., 2019). A fine-tuning technique which involves adapting a pre-trained LLM to a specific domain or task by training it further on a smaller, task-specific dataset. This technique enables the model to answer more specialized queries and handle domain-specific tasks more effectively (Raffel et al., 2020). However, fine-tuning can be resource-intensive, requiring substantial computational power, large labelled datasets, and repeated retraining to maintain up-to-date knowledge (Hadi et al., 2023). This makes it an expensive and sometimes impractical solution, especially in rapidly evolving fields where information changes frequently. To mitigate these challenges, RAG has emerged as a promising alternative (Lewis et al., 2020). RAG enhances LLM performance by integrating external data sources, allowing the model to retrieve relevant information dynamically at the time of inference. This method provides access to up-to-date and domain-specific knowledge without the need for extensive fine-tuning, improving the model's accuracy and reducing the computational cost (W. X. Zhao et al., 2023). Thus, by retrieving relevant documents or knowledge bases, RAG enables LLMs to handle knowledge-intensive tasks more effectively, making it an ideal solution for applications where the information landscape is constantly evolving (Naveed et al., 2023).

Therefore, organizations are increasingly interested in leveraging LLMs to enhance their operations by providing more accurate and contextual relevant information (Bhat et al., 2024). Particularly, in customer service and support chatbots are increasingly used to reduce human support personnel and to overcome the limited availability of domain-experts (Shuster et al., 2021). The overall goals are cost-savings, redirection of valuable human resources to other important tasks, increased

availability of the service and overall customer satisfaction (Afzal et al., 2024; Bhat et al., 2024). Given these advantages, it is no surprise that the global chatbot market, valued at 5.39 billion dollars in 2023, and is expected to reach 42.83 billion dollars by 2033 (Spherical Insights & Consulting, 2024). With the breakthrough of LLMs which are equipped with billions of parameters and pre-trained on vast language datasets, new possibilities for conversational agents are offered (J. Wei et al., 2024). By crafting specific prompts, LLMs can generate human-like conversational responses without the need for training data, effectively functioning as chatbots (J. Wei et al., 2024). In contrast to other frameworks, LLMs demonstrate significant potential in supporting chatbots that are context-aware and capable of responding to off-topic user messages (Volum et al., 2022). Therefore, the approach of enhancing LLMs with RAG seems like a promising solution to build chatbots for Customer Support. Especially, in technical support RAG should be suitable because questions are highly specific from a wide range of problems, very repetitive, and sequentially solvable. Thus, this master's thesis will test the hypothesis that enhancing LLMs with RAG is suitable for Customer Support Chatbots by building a Chatbot for the IT Support of the University of Münster.

The Center for Information Technology (CIT) of the University of Münster provides IT services mainly for their students and staff members. Via Web they provide guides, FAQs and documentations on different IT services including access to the University's Internet, communication services, software applications, how to use the IT infrastructure (e.g. Remote Desktop, Uni Cloud, etc.) and technical support for teaching, research, and administrative purposes. Beyond that they provide User Support during the day in a form of an IT Hotline, Service Desk, and an on-site IT Support. With the increasing number of provided services the Web page became cluttered and complex to use, therefore more questions are sent to the User Support which could have been answered in the documentation on the Web already. Thus, the overall goal is to tackle this problem by leveraging a RAG-based chatbot to complement the User Support to substitute the overwhelming navigation on the web page and to improve the availability of the User Support. Further, this use case is suitable to answer the stated hypothesis as the University of Münster hosts their own on-premise LLMs where it is ideal to build a RAG-System with. Currently, the on-premise LLMs, namely Llama 3.1-70B and Mixtral 8x7B, are in use for their UniGPT chatbot service which was developed to provide a GPT service for dealing with sensitive data and projects where data protection is paramount. Therefore, an on-premise LLM is especially advantageous when dealing with Customer requests which contain sensitive data that needs to be processed. Additionally, they provide up-to-date Large Language Model (LLM)s by OpenAI, such as gpt-4o-mini. Lastly, with

over 46,000 students, the University of Münster ranks among the largest universities in Germany. Given the resource-intensive nature of IT support for such a large institution, the introduction of a complementary chatbot could significantly reduce operational costs while enhancing service efficiency.

Based on the problem statement that LLM are limited concerning domain specific inquiries and the hypothesis that enhanced LLMs with RAG are suitable for building effective chatbots in customer support, the research objective of this thesis is twofold: First, an implementation of a RAG-based chatbot LLM-agnostic for the University of Münster with the goal to improve their IT support by reducing user dependency on web navigation and improving access to specific information. Second, to rigorously evaluate this chatbot's performance in meeting the unique demands of an IT support environment. The evaluation will focus on the chatbot's ability to handle domain-specific queries, improve user experience, and optimize support operations. For that the evaluation will be based on user surveys, feedback from IT staff, and a structured performance test. Through these artifacts, this thesis aims to contribute insights on the feasibility and practical benefits of RAG-enhanced LLMs within institutional support frameworks, ultimately testing whether this approach can improve customer service in complex technical support scenarios.

The remainder of this thesis is structured as follows: Chapter 2 provides the theoretical background of chatbots, LLMs, RAG, and case studies of chatbots. Chapter 3 presents the methods used, including the use of the Design Science Research (DSR) Methodology to guide artifact development and a mixed-method approach for evaluation. Chapter 4 describes the implementation of the RAG-based chatbot, including technical architecture, system logic and deployment. Chapter 5 details the evaluation process and presents the results from user studies, staff feedback, and performance assessments. Chapter 6 contains a critical discussion of the results in light of the research objectives, followed by Chapter 7, which concludes the thesis and outlines implications for future work.

2 Background

This chapter provides the theoretical foundation and contextual knowledge necessary to understand RAG-based chatbots and their application in domain specific environments. It begins with an overview of conversational Artificial Intelligence (AI) and chatbot technologies, followed by an introduction of Large Language Models and recent developments in the field of NLP. After that, the chapter examines RAG in detail, including its architecture, advantages over alternative approaches, optimization techniques, and best practices from the literature. The chapter concludes by addressing the application of RAG-based chatbots in domain-specific business contexts, discussing their benefits, implementation challenges, and key deployment considerations for enterprise environments.

2.1 Chatbots: An advancement in Human-Computer Interaction

The way humans interact with computers has evolved significantly over the decades, driven by advancements in Human-Computer Interaction (HCI). HCI *"is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them"* (Sinha et al., 2010). Traditionally, HCI was limited to command-line interfaces and graphical user interfaces (GUIs), requiring users to adapt to system structures (Sinha et al., 2010). However, with the rise of AI, HCI has become more natural and intuitive, allowing users to engage with computers using natural language rather than complex commands (Ramírez, 2024). A key innovation in this transformation is the development of chatbots. According to the dictionary, a chatbot is *"a computer program designed to respond with conversational or informational replies to verbal or written messages from users."* (Dictionary.com, 2025). Thus, a chatbot is an AI program and a HCI model, which facilitates dialogue between users and machines via text or voice-based exchanges (Adamopoulou & Moussiades, 2020; Bansal & Khan, 2018). Here, AI program refers to the utilization of NLP by chatbots when evaluating the input of the users and when generating the response by the system (Corea, 2019). NLP is an interdisciplinary field at the intersection of linguistics, computer science, and AI. NLP enables the computer to comprehend, interpret and generate human language in a manner that imitates human communication (Ramírez, 2024). Therefore, the application of NLP is fundamental for the development of chatbots.

2.1.1 General Chatbot Architecture

Suhaili et al. (2021) present a general chatbot pipeline by introducing the most relevant components, each performing essential functions to build a chatbot system (see Fig. 1). The first step is the NLP Pre-Processing of the text input. In this step, techniques like tokenization, lemmatization, and stemming to the user's query are applied to convert the text into structured data. The next step is Natural Language Understanding (NLU) where user input is analysed by identifying intent and extracting relevant details. The Dialogue Manager processes the previous structured user input, maintains the dialogue context, and decides the system's next action. If information is missing or unclear, it can request clarification to ensure semantic accuracy. Data Sources store and retrieve information that the Dialogue Manager utilizes to generate responses. These sources can be internal, such as predefined rules or databases, or external, accessed through third-party services like Web APIs to fetch relevant information. The Response Generator selects the most appropriate response from a set of candidate options or generates dynamically a response based on the processed input.

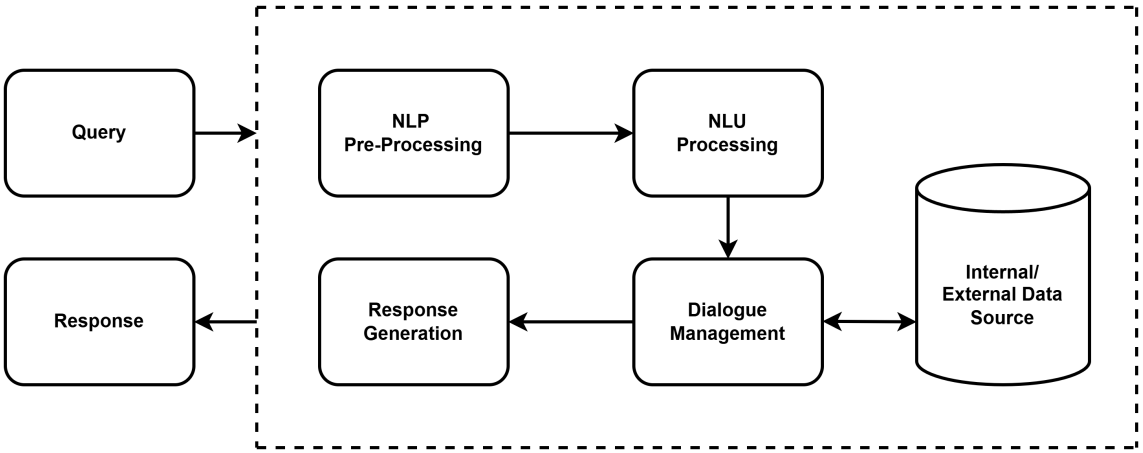


Figure 1 General Chatbot Pipeline (Suhaili et al., 2021)

2.1.2 Chatbot Design Principles

To develop a chatbot there are several design principles. A quick look into recent literature shows that there exist different names for the design approaches even though they describing the same principle. Singh and Namin (2025) distinguish the chatbot designs between rule-based and AI-based approaches. The rule-based approach relies on predefined patterns and rules to generate responses based on user input (Adamopoulou & Moussiades, 2020). AI-based approaches are majorly classified into retrieval-based approaches and generative-based approaches. Chatbots that

follow the retrieval-based approach typically search a pre-existing dialogue database to select the response that best matches the users' queries (Bartl & Spanakis, 2017; Z. Wang et al., 2019).In generative-based approaches, AI model chatbots produce responses one word at a time, forming completely new sentences tailored to the user's queries (Singh & Namin, 2025). Adamopoulou and Moussiades (2020) differentiating between pattern matching approaches and machine learning approaches. Where Adamopoulou and Moussiades (2020) use pattern matching and rule-based approach interchangeably. And machine learning approaches are also classified into retrieval and generative-based approaches. Lastly, Suhaili et al. (2021) divide the chatbot design approaches into retrieval-based and generative-based approaches. Where rule-based Chatbots are defined as a sub-category of retrieval-based chatbots. For this reason, the rule-based, retrieval-based, and generative-based approaches are presented more in detail below, where retrieval-based and generative-based approaches are examined from the machine learning perspective.

2.1.3 Dialogue in Chatbot Design

Before diving into the design approaches, it is crucial to first understand the different types of dialogues in HCI and provide a formal definition of a dialogue system. Thus, when examining dialogue, several key dimensions help categorize conversational patterns. As Ramesh et al. (2017) explains, conversations can be classified based on their length into two primary types. Short-text conversations produce a single response to a single input, such as answering a specific question queried by a user. In contrast, long-text conversations involve extensive information exchange where historical context must be maintained to generate appropriate responses, e.g. by customer care conversations with multiple questions and responses. Another critical distinction lies between open-domain and closed-domain (or domain-specific) conversations. According to Ramesh et al. (2017), open-domain conversations mirror natural human interactions where topics may shift across domains over time, e.g. similar to exchanges on social media platforms. These conversations are challenging to automate due to the extensive knowledge required to generate reasonable responses across various topics. In contrast, closed-domain conversations operate within limited knowledge boundaries specific to particular domains, such as customer support, health care service, e-learning, etc., where interactions remain focused on addressing specific purposes without deviating from the intended domain (Ramesh et al., 2017).

The literature shows a well-defined problem formulation for dialogue systems. First, a dialogue *session* refers to a complete conversational exchange. If a session consists

of only one interaction between two participants, comprising a *query* and a *response*, it is classified as a *single-turn dialogue* (Shang et al., 2015). On the other hand, if multiple exchanges occur within the session, it is considered a *multi-turn dialogue* (Jurafsky, 2000). In such cases, the final utterance serves as the *response*, while the preceding exchanges form the *context*. Lastly, when a conversation involves only two participants, it is known as a *dyadic dialogue*. And if more than two individuals are engaged, it is referred to as a *multi-party dialogue* (Hsueh et al., 2006).

In addition to conversational data, external resources such as structured documents, knowledge graphs, or unstructured text can support dialogue generation (Fu et al., 2022). In multi-modal settings, inputs like images, videos, or adios may also be used, depending on the task (Fu et al., 2022).

In essence a dialogue system's goal is to return a response to a query from a user, while taking the conversational data meaning the chat history and external sources if available into account. Therefore, Fu et al. (2022) formally describe a dialogue system as follows: Given a dialogue context $C = X_1Y_1X_2Y_2 \dots X_{t-1}Y_{t-1}$ and the last query X_t as an input from one user, a model should generate an appropriate response Y_t, utilizing external resources S. Thus, a dialogue system can be expressed as:

$$\hat{Y} = \arg\max_{Y \in \Omega} P(Y \mid C, S)$$

where Ω represents the search space for possible responses, and S serves as the external knowledge base to support the dialogue.

In a retrieval-based dialogue system, the search space consists of a predefined set of candidate responses $Y_1, Y_2, Y_3, \dots, Y_n$, where n is typically equal or proportional to the dataset size. In contrast, a generation-based system has a search space that grows exponentially with the vocabulary size $|\mathcal{V}|$ and the allowable response length range $[l_{\min}, l_{\max}]$. At the core of the model lies a search and scoring function P, which navigates the hypothesis space to identify the most suitable response $\hat{Y}$. Therefore, search spaces differ for different models and approaches, for retrieval-based and generative-based approaches, more details are provided in their respective subsection.

2.1.4 Rule-based Chatbots

Rule-based chatbot design relies on predefined patterns and rules to generate responses based on user input (Adamopoulou & Moussiades, 2020). These chatbots use pattern-matching algorithms to compare user queries with stored answers, se-

lecting the most appropriate response from a predefined set (Hussain et al., 2019). Context can also influence rule selection, allowing for slightly more dynamic interactions with the user (Marietto et al., 2013). The main advantages of this approach include fast response times and ease of implementation, as no complex learning models are required (Jia, 2009; Singh & Namin, 2025). However, rule-based chatbots struggle with grammatical and syntactic variations in user input, leading to rigid and sometimes repetitive responses that lack human-like spontaneity (Ramesh et al., 2017). Further, maintaining and expanding the rule set is labour-intensive, as thousands of rules may be needed for a chatbot to function effectively (Adamopoulou & Moussiades, 2020).

Inspired by the Turing test proposed by Alan Turing in 1950, researchers have developed various rule-based chatbots to simulate human-like conversations to convince users that they are chatting with a real person (Shieber, 1994; Shum et al., 2018; Turing, 2009). One of the earliest examples is Eliza, created by Joseph Weizenbaum in 1966, which used hand-crafted scripts to mimic a Rogerian psychotherapist through pattern matching, despite lacking true understanding of the conversation (Weizenbaum, 1966). Kenneth Colby developed Parry in 1972, that introduced a more advanced structure by incorporating a mental model that simulated paranoia, allowing it to adjust its responses based on emotional intensity (Colby et al., 1972). In addition to the pattern matching algorithm, Colby has built in a parsing module, which attempts to understand the interview by extracting meaningful information from the input and then feeding it as input to the interpretation module, where the pattern matching then takes place (Colby, 1981). Later, Alice, introduced by Richard Wallace, utilized Artificial Intelligence Markup Language (AIML) to enable recursive pattern matching, making chatbot customization more accessible (Wallace, 2009). Although Alice demonstrated improvements in chatbot design, it still struggled with long-term dialogue coherence, which prevented it from passing the Turing test (Shieber, 1994; Shum et al., 2018). These chatbots illustrate the gradual advancements in rule-based systems, from simple scripted responses to more dynamic conversational models, yet they remain limited by predefined patterns and lack true conversational understanding (Shum et al., 2018).

2.1.5 Retrieval-based Chatbots

Retrieval-based methods in chatbot design operate on the principle that the appropriate dialogue response exists within a predefined set of candidate responses (Fu et al., 2022). As Fu et al. (2022) explain, these methods evaluate all possible replies in a candidate set, assigning each a score to determine its appropriateness

for the current context, ultimately selecting the highest-scoring candidate as the output response. The core components of a retrieval-based model include a score function *s()* which determines the matching score for context-response pairs, and an encoding function *e()* that converts natural language context and responses into dense representations (Fu et al., 2022). According to Huang et al. (2020), existing retrieval-based methods can be categorized into "shallow interaction" and "deep interaction" based on the forms of these encoding and score functions.

In *shallow interaction*, as Huang et al. (2020) elaborate, candidate responses and queries are encoded independently. After the encoding phase the best possible answer is found by the highest scoring pair of context and response, which can be formulated as:

$$\hat{Y} = \arg\max_{Y \in \mathcal{Y}} s(e(C), e(Y))$$

where $\mathcal{Y}$ represents the candidate set (Fu et al., 2022). Early retrieval-based methods used shallow interaction and primarily relied on lexical co-occurrence or syntax analysis (Kang et al., 2014; M. Wang et al., 2015; Z. Wang et al., 2017). Common scoring techniques include TF-IDF (Term Frequency - Inverse Document Frequency), the number of common words, and using dependency trees (Kang et al., 2014; M. Wang et al., 2015; Z. Wang et al., 2017). With the rise of neural networks and deep learning, modern approaches increasingly use neural scoring functions, such as bilinear functions, multi-layer perceptrons, or Euclidean distance (Hu et al., 2014; Lu & Li, 2013; Yang et al., 2018). The encoding function could be handled by a Convolutional Neural Network (CNN), a Recurrent Neural Network (RNN), or a combination of both (Fu et al., 2022).

A retrieval-based method is considered as a *deep interaction* if the interaction between the dialogue context and a candidate response begins during the encoding phase, such that their representations are computed with mutual awareness (Fu et al., 2022). In this case, the scoring function is defined as:

$$\hat{Y} = \arg\max_{Y \in \mathcal{Y}} s(e(C, Y))$$

Here, the key difference lies in the encoding function $e(\cdot, \cdot)$, which encodes the context and response jointly (Fu et al., 2022). Many deep interaction models treat multi-turn dialogues as sequences, with each utterance as an element (Y. Wu et al., 2016; Yang et al., 2018; X. Zhou et al., 2018). The overall matching between the dialogue context and a candidate response is then broken down into the matching between each individual utterance and the response (Fu et al., 2022).

Retrieval-based approaches also rely on predefined responses, similar to rule-based methods(C.-W. Liu et al., 2016). However, the key distinction lies in their use of more advanced techniques, such as machine learning algorithms or semantic similarity measures for the scoring function, which draw on information from databases or knowledge bases rather than relying on human-made rules (C.-W. Liu et al., 2016; Suhaili et al., 2021). Suhaili et al. (2021) highlight that since retrieval-based chatbots use pre-defined responses, they produce no grammatical errors, requiring only sufficient intelligence to match requests with appropriate responses. However, these benefits come with constraints, Caldarini et al. (2022) point out that creating a knowledge base for retrieval-based approaches can be expensive and time-consuming, requiring extensive training and a comprehensive knowledge base, while their simplicity in design limits their ability to handle advanced queries.

2.1.6 Generative-based Chatbots

In contrast to retrieval-based methods, which rely on selecting the most suitable response from a predefined set, generative-based approaches create responses dynamically (Fu et al., 2022). In this paradigm, the chatbot generates text word by word, constructing entirely new sentences based on the user's input (Singh & Namin, 2025). Generative-based chatbot systems are powered by deep learning models that learn linguistic patterns and semantics from large corpora of text, common algorithms include RNN and Long Short-Term Memory (LSTM) (Caldarini et al., 2022). While these models laid the foundation for generative dialog systems, they often suffer from limitations such as generating low-quality or inconsistent responses (Hussain et al., 2019).

Generative dialogue systems are typically implemented using an encoder-decoder architecture, a foundational structure in sequence-to-sequence learning models (Sabry, 2023). In this architecture, the encoder takes the user's input, typically a sentence or sequence of words, and converts it into a fixed-length internal representation known as a context vector or hidden state (Cho et al., 2014; Murel & Noble, 2024). This vector serves as a so called dense summary or representation of the semantic and syntactic information in the input, capturing the meaning of the entire sentence (Fu et al., 2022). The decoder then uses this context vector to generate a response (Cho et al., 2014). It does so sequentially, producing one unit of text at a time (Fu et al., 2022). Each unit—called a token, it can represent a word, subword, or even a single character depending on the tokenizer used by the model. For example, the word "ChatGPT" is split by the GPT-4o model into the tokens "Chat" and "GPT", while a more fine-grained tokenizer might split it into smaller parts like "Ch", "at",

"G", "PT" (OpenAI, 2024d). The decoder begins by generating the first token of the response based on the context vector. Then, it uses both the context vector and the previously generated tokens to produce the next token. This continues until an end-of-sentence token is generated or a predefined maximum length is reached. This token-by-token generation process is why the method is called auto-regressive, meaning the generation of each token depends on the tokens that came before it.

Formally, the goal of a generative dialogue system is to find the most probable response $\hat{Y}$ from the space of all possible responses Ω. The model aims to maximize the joint probability of all individual tokens in the response, given the user's input C and the tokens that have already been generated:

$$\hat{Y} = \arg\max_{Y \in \Omega} \prod_{i=1}^{|l_Y|} p_\theta(y_i \mid C, y_{<i})$$

Here, p_θ denotes the model's conditional probability function parameterized by θ, $|l_Y|$ is the length of the response Y, y_i is the i-th token in the response, and $y_{<i}$ represents all generated tokens before the i-th timestep (Fu et al., 2022). Although this method tends to produce high-quality and coherent responses, it suffers from slow inference speeds because of its sequential nature. Because each word must be generated in order, parallelization is not possible during the generation phase. To address this limitation, researchers have proposed non-auto-regressive generation techniques, which assume that each token in the response is conditionally independent from others given the input context. This allows the model to generate all tokens in parallel, significantly speeding up the response time:

$$\hat{Y} = \arg\max_{Y \in \Omega} \prod_{i=1}^{|l_Y|} p_\theta(y_i \mid C)$$

(Fu et al., 2022). Although non-auto-regressive methods are faster, they currently lag behind in terms of the quality and coherence of the generated responses Kaiser et al., 2018; Lee et al., 2018. Nonetheless, they open promising avenues for real-time conversational systems and remain an active area of research.

A major breakthrough in generative approaches was the introduction of the Transformer architecture by Vaswani et al. (2017). Unlike earlier approaches based on RNNs or CNNs, the Transformer dispenses both recurrence and convolutions entirely. Instead, it relies solely on a mechanism called self-attention to model dependencies between words (Vaswani et al., 2017). On the one hand, RNNs process

sequences by maintaining a hidden state that is updated at each time step (Salehinejad et al., 2017). This allows them to handle variable-length inputs and outputs, but they suffer from problems like vanishing gradients and difficulty modelling long-range dependencies (Salehinejad et al., 2017). CNNs, on the other hand, capture local features in sequences through convolutional filters, but struggle to model global sequence structure unless stacked deeply (Yin et al., 2017).

The Transformer architecture addresses these limitations by using self-attention to capture relationships between all words in a sequence, regardless of their positions. This enables parallel processing of tokens during training and more effective learning of contextual information (Vaswani et al., 2017). Although the Transformer is also auto-regressive during generation, meaning predicting one token at a time based on the preceding ones, it differs from RNNs in how it processes information. Unlike RNNs, which must compute each step in strict sequence (token by token), the Transformer allows parallel computation during training by attending to all positions in the input at once using self-attention (Salehinejad et al., 2017; Vaswani et al., 2017). This removes the constraint of processing tokens strictly one after the other and significantly improves efficiency (Vaswani et al., 2017). Due to its superior performance, scalability, and training efficiency, the Transformer architecture has become the de facto standard in natural language processing (Hadi et al., 2023; Vaswani et al., 2017). Today, most state-of-the-art LLMs, including GPT, Claude, Gemini, and LLaMA and many more, are based on variants of the Transformer architecture (Hadi et al., 2023). This paradigm shift paved the way for the development of LLMs, which are Transformer-based models trained on massive text corpora comprising billions of tokens. These models are capable of understanding and generating human-like text across a wide range of tasks, including question answering, summarization, translation, and dialogue (Hadi et al., 2023). The following section will dive deeper into Large Language Models, providing an overview of their tasks, applications, and the key challenges and limitations they face.

2.2 Large Language Models

LLMs represent a transformative leap in NLP, often described as “next-generation” or “transformative” due to their significant impact on the field (Hadi et al., 2023; J. Wei, Tay, et al., 2022). These models are primarily built upon the Transformer architecture, a deep learning innovation introduced by Vaswani et al. (2017) in their landmark paper *"Attention is All You Need"*. Unlike earlier models, the Transformer's self-attention mechanism enables the efficient modelling of long-range dependencies and parallel processing during training, which has made it possible to

scale models massively across multiple GPUs (Carion et al., 2020; Shaw et al., 2018). This foundation has led to the development of some of the most powerful LLMs, such as OpenAI's GPT series (Ghojogh & Ghodsi, 2020), Google's BERT and Gemini (Devlin et al., 2019; Pichai & Hassabis, 2023), Claude from Anthropic (Anthropic, 2023), Llama by Meta (Touvron et al., 2023), and many other models that revolutionized the way machines process and generate human language. One of the core strengths of LLMs lies in their ability to ingest and process vast amounts of unstructured text data, capturing semantic relationships between words and phrases (Adnan & Akbar, 2019; Dong et al., 2023). Furthermore, many of these models have been extended to handle not only textual inputs but also multi-modal data, such as visuals and audio, trained to understand semantic relationships between them (Awais et al., 2025). The pre-training and fine-tuning paradigm has become central to their training methodology. Models are initially exposed to massive internet text corpora to acquire general language understanding, and then fine-tuned on specific datasets to adapt to particular tasks (Hadi et al., 2023; Naveed et al., 2023). For instance, ChatGPT, LLaMA, and Falcon, which are all variants of the Generative Pre-trained Generative Pre-trained (GPT) Transformer model, have been fine-tuned for diverse applications, ranging from conversational agents to specialized research tools (Chiang et al., 2023; Katz et al., 2024; Zhuo et al., 2023). The release of GPT-1 in 2018 marked the beginning of this new era, with its 117 million parameters showcasing the potential of transformer-based architectures in generating coherent text (Akbar et al., 2021; Radford et al., 2018). Subsequent iterations, such as GPT-3 in 2020 and GPT-4 thereafter, demonstrated the ability of a variety of NLP tasks and even improved the ability coherency and natural sounding (Floridi & Chiriatti, 2020; Hadi et al., 2023). Major tech companies joined the AI race, introducing their own models, e.g. Meta's LLaMA (Touvron et al., 2023), Google's Gemini (Pichai & Hassabis, 2023), Amazon's Alexa-enhanced AI (Khatri et al., 2018), and Huawei's Pangu (Zeng et al., 2021). Today, LLMs have become foundational to modern AI systems, enabling capabilities such as dialogue generation, summarization, translation, and beyond (Hadi et al., 2023).

2.2.1 Large Language Model Tasks

LLMs have demonstrated remarkable capabilities across a wide range of NLP tasks. Hadi et al. (2023) published a comprehensive overview of LLMs presenting a taxonomy of LLMs tasks: In the domain of question answering, LLMs leverage their semantic understanding to accurately extract or generate responses from complex textual contexts (Su et al., 2019). Their ability to produce coherent and contextually appropriate output also makes them highly effective in text generation for

applications such as article writing, social media posts, and source code creation (Celikyilmaz et al., 2020; Zhu et al., 2023). In language translation, LLMs provide fluent and accurate translations, enabling seamless communication across linguistic boundaries (L. Wang, Lyu, et al., 2023). Their text classification capabilities support efficient organization of large-scale textual data, facilitating sentiment analysis, spam detection, and content moderation (Labonne & Moran, 2023; Mullick et al., 2023; W. Zhang et al., 2023). Moreover, LLMs are proficient in summarization, condensing lengthy documents into concise texts while preserving key information (T. Zhang et al., 2024). As previously told, LLMs also play a pivotal role in powering virtual assistants and chatbots through natural and responsive conversational interactions, while reducing workload of human personnel, showing again the potential of improved HCI and customer support and service (Hadi et al., 2023; Luo et al., 2023; Susanto & Khaq, 2024). In information extraction, LLMs enable the identification of entities, relationships, and events from unstructured text, contributing to the construction of structured knowledge bases (X. Wei et al., 2023). In the area of semantic search, LLMs interpret user intent beyond keyword matching, improving the relevance and accuracy of retrieved information (Zhuo et al., 2023). Finally, LLMs have advanced the field of automated speech recognition by providing robust transcription capabilities, even in the presence of accents or domain-specific vocabulary (Radford et al., 2023). Collectively, these functionalities underscore the transformative impact of LLMs in advancing language-based AI systems for HCI.

2.2.2 Applications of Large Language Models

Given their ability to perform a wide range of tasks, LLMs are increasingly being adopted across various domains and industries. Hadi et al. (2023) and Naveed et al. (2023) presenting a variety of applications in various fields: In the medical domain, LLMs demonstrate significant potential across various applications, including medical education, clinical decision-making, radiologic and genetic analysis, patient communication, medical imaging interpretation, and drug discovery, thereby enhancing both patient care and healthcare delivery systems (Hadi et al., 2023; Naveed et al., 2023). In education, LLMs are being leveraged to personalize learning, assist with assignments and feedback, support teaching, and automate grading. Although their effectiveness, potential drawbacks include decline in students' creativity and critical thinking skills, the risk of inadequate training and contextual fine-tuning limiting their effectiveness, and fears of job displacement for educators, potentially deepening educational inequality (Hadi et al., 2023; Naveed et al., 2023). In the finance sector, LLMs are being applied to financial tasks such as risk assessment, market prediction, and algorithmic trading, offering enhanced decision-making (Hadi et al., 2023;

Naveed et al., 2023). In engineering, LLMs are increasingly being utilized for tasks such as code generation, debugging, design support, and troubleshooting, although challenges remain regarding the accuracy of complex technical computations (Hadi et al., 2023; Naveed et al., 2023). In the media and entertainment industry, LLMs are revolutionizing content creation, personalization, and audience engagement by enabling automated media generation, personalized recommendations, and virtual presenters, thereby transforming how content is produced, curated, and consumed (Hadi et al., 2023). LLMs help to transform the legal profession by enhancing legal research, drafting, and decision-making, with tools like Chatlaw (Cui et al., 2023). Nonetheless, there are observed inaccuracies in legal citations, which highlight the need for careful integration into legal practice (Carrick & Kesteven, 2023). Lastly, LLM-based chatbots and virtual assistants are transforming customer service by offering scalable, efficient, and instant responses to customer inquiries (Hadi et al., 2023; Olujimi & Ade-Ibijola, 2023). Unlike human staff, they can handle large volumes of interactions simultaneously, reducing wait times and boosting customer satisfaction (Howell et al., 2023). This automation also lowers costs by reducing the need for large support teams (Allen et al., 2023). Thus, an increasing number of companies employing NLP to support human staff instead of employing more expensive human personnel (Olujimi & Ade-Ibijola, 2023). This eases the workload while maintaining timely and personalized service. In IT support, LLMs can provide 24/7 help with common issues, allowing human staff to focus on more complex tasks (Susanto & Khaq, 2024).

2.2.3 Limitations and Challenges of Large Language Models

While LLMs have made notable advancements in various fields, they still face significant limitations and challenges (Hadi et al., 2023; Naveed et al., 2023). Although LLMs are often viewed as forerunners of Artificial General Intelligence (AGI), their impressive performance in conversational tasks doesn't fully address the many areas in which they fall short, making them unlikely to represent an early form of AGI (Hadi et al., 2023). For example, LLMs face significant challenges in terms of training data, as the vast size and complexity of datasets make them prone to biases, duplicates, and privacy leaks, while also complicating quality assessment and accurate evaluation due to the black-box nature of data distribution and unclear data requirements for specific tasks (Kaddour, 2023; Kaplan et al., 2020). Tokenization in LLMs involves breaking down words into smaller units, often using subword tokenization to handle unfamiliar vocabulary while managing computational complexity (Sennrich et al., 2015). However, this method can lead to issues like inconsistent token combinations affecting API pricing (Hadi et al., 2023), as well as

unexpected model responses in multilingual settings, such as in Mandarin variations due to different spacing in prompts (Fujii et al., 2023). Training LLMs incurs high computational costs, both financially and environmentally, requiring vast resources, and scaling these models poses further challenges, prompting the introduction of Computer Optimal Training to enhance efficiency (Kaplan et al., 2020; Schwartz et al., 2020; Strubell et al., 2020). Again, Fine-tuning LLMs for custom tasks needs a high amount of memory and computational resources, limits accessibility, and contribute to high carbon emissions (Hadi et al., 2023; Schwartz et al., 2020; X.-K. Wu et al., 2025). Another major challenge of LLMs is high inference latency, primarily caused by large memory footprints and lack of model parallelism (Hadi et al., 2023). Further, a key challenge of LLMs is the limited context length, which affects their ability to interpret prompts and perform semantic analysis effectively (Hadi et al., 2023). Next challenge for LLMs is knowledge updating and refinement, as the knowledge acquired during pre-training is limited and can become outdated over time (Naveed et al., 2023). Yet retraining the model with updated data is costly and not a sustainable solution (Hadi et al., 2023). Lastly, Hadi et al. (2023) describe the risk of foundation models, where a foundation model refers to a base model that serves for various NLP tasks. For example following risks are listed by Hadi et al. (2023): Bias, which can arise from biased training data, user inputs, or algorithmic design itself, leading to unfair or discriminatory outputs. Further, information hallucination occurs when models generate plausible-sounding but factually incorrect content due to gaps in knowledge or context. Lack of explainability makes it difficult to understand or justify the model's outputs, which raises concerns around trust and accountability. Reasoning errors highlight limitations in logic, planning, and common sense, often causing flawed conclusions. Security vulnerabilities, such as prompt injection, jailbreaks, and data poisoning, can manipulate model behaviour in harmful ways. Adversarial attacks exploit model weaknesses by subtly altering inputs to trigger incorrect or dangerous responses. Behavioural changes over time, also referred as model drift, may lead to unpredictable performance fluctuations. Lastly, foundation models may struggle with basic tasks like spelling and counting due to their statistical rather than symbolic nature.

2.2.4 Countermeasures for expensive Fine-Tuning

In response to these challenges, numerous techniques and improvements have been proposed to enhance the performance and reliability of LLMs. For example, in the case of Fine-tuning a model on domain-specific tasks there are three prominent alternative approaches proposed namely prompt engineering, knowledge distillation and RAG (X.-K. Wu et al., 2025). Prompt engineering is to provide the LLM a set

of instructions to enhance or refine the models behaviour and capabilities. A prompt can be used to influence the conversation with an LLM providing specific rules and guidelines (White et al., 2023).Further, Knowledge distillation is a technique used to transfer the capabilities of a large, complex model to a smaller, more efficient one, making it especially valuable in scenarios with limited computational resources (X.-K. Wu et al., 2025). While fine-tuning adapts a model to specific tasks using domain-relevant data to enhance its performance, knowledge distillation focuses on compressing the model by transferring learned patterns and behaviours (X.-K. Wu et al., 2025). Lastly, RAG improves the models capability by retrieving relevant information from external knowledge bases. The relevant external information is then fed with the query into the LLM to improve the generated response. This type of hybrid approach combining a retrieval-based and generative-based approach has emerged as a cheaper solution for fine-tuning LLMs for domain-specific tasks instead of expensive fine-tuning the whole model. (S. Wu et al., 2024). Furthermore, RAG emerged as a viable solution to mitigate against more limitations of LLMs. For example, RAG can reduce hallucinations by filling knowledge gaps, mitigate against outdated data and information by retrieving up-to-date data and improve the models accuracy by more relevant and concise information retrieval (Lewis et al., 2020). A whole new research area emerged to leverage these potential advantages, resulting in a wide range of applications of RAG. Arslan et al. (2024) present over 50 studies which apply the RAG technology for various LLM tasks. Up to 20 studies were dedicated to questions-answering tasks underlying the potential for RAG in domain-specific chatbot systems.

2.2.5 Applications of Retrieval-Augmented Generation

When looking into recent literature a wide range of RAG-based chatbots have been developed. For example, Bhat et al. (2024) built a RAG-based restaurant chatbot, which is able to interact with customers in natural language, to improve the dining industry. Arabi et al. (2024) implemented Habit Coach, a chatbot which supports you to build habits on a daily-basis. Another related work is by Antico et al. (2024) who developed a chatbot for the University of Milano-Bicocca, which is able to retrieve university-related documents and links for students. Quidwai and Lagana (2024) developed a RAG-based chatbot for personalized treatment recommendations for Multiple Myeloma patients. This chatbot is able to retrieve relevant Multiple Myeloma literature based on patient-specific genomic data. Further, Vakayil et al. (2024) implemented a chatbot assisting victims of sexual harassment based on the LLM Llama-2 model, which is enhanced with RAG to improve the accuracy of providing information in an understanding and helpful tone by retrieving sources from

the government or legal system of India. Moreover, Gamage et al. (2024) build a multi-agent RAG chatbot architecture for decision support in net-zero emission energy systems. For the RAG logic Gamage et al. (2024) implemented four agents: the Observer, which monitors energy environments for key events; the Behavior Analyzer, which interprets energy patterns for behavior analysis; the Visualizer, which creates intuitive visualizations; and the Knowledge Retriever, which extracts and compiles relevant documents and images. These agents working together to retrieve the relevant context which was afterwards processed by an LLM. Again, another work developed a RAG-based chatbot for students and staff in the university context: Olawore et al. (2023) developed a chatbot for the Ulster University in Northern Ireland leveraging information of websites about different faculties, including details on fees, courses, modules, and assessment.

These examples highlight the growing trend of RAG-based chatbots tailored to specific domains, demonstrating their potential to enhance user interaction and support in various sectors. Building on this development, the central aim of this master thesis is to contribute to the field by designing, developing and evaluating a RAG-based chatbot specifically for the university IT support of Muenster. The focus lies on investigating whether such a chatbot can effectively assist IT staff in their daily operations and whether it is likely to be adopted by students as a helpful support tool. To lay the foundation for this, the following section provides a deeper exploration of RAG, introducing its architecture and outlining best practices for implementation.

2.3 Retrieval-Augmented Generation (RAG)

Lewis et al. (2020) were the first to introduce the term Retrieval-Augmented Generation to describe a hybrid approach that combines generative language models with a retrieval-based mechanism. Their core idea was to improve the factual accuracy and scalability of generation models by incorporating relevant external documents at inference time, rather than relying solely on parametric knowledge stored within the model's weights. In their formulation, RAG consists of two main components: a retriever that selects relevant documents from a large corpus based on the input query, and a generator that conditions on both the query and the retrieved documents to produce an output. The retriever is typically based on dense vector representations, while the generator in the original work is a sequence-to-sequence model, specifically BART. Lewis et al. (2020) propose two variants of RAG, namely RAG-Sequence and RAG-Token:

In the **RAG-Sequence** model, each retrieved document is treated as a latent variable for the entire output sequence. The model marginalizes over the top-k retrieved documents to compute the sequence likelihood:

$$p_{\text{RAG-Sequence}}(y \mid x) \approx \sum_{z \in \text{top-}k} p_\eta(z \mid x) \cdot p_\theta(y \mid x, z) \tag{2.1}$$

Here, x is the input query, z is a retrieved document, and y is the generated output.

The **RAG-Token** model, on the other hand, allows different documents to influence different tokens during generation. The model marginalizes over the retrieved documents at each token position:

$$p_{\text{RAG-Token}}(y \mid x) \approx \prod_i \sum_{z \in \text{top-}k} p_\eta(z \mid x) \cdot p_\theta(y_i \mid x, z, y_{1:i-1}) \tag{2.2}$$

This allows the model to dynamically switch between sources at the token level, enabling more flexible integration of retrieved information.

With the rise of LLMs, the implementation of RAG has evolved significantly. In modern RAG systems, such as those built with GPT-based models, the retrieved documents are no longer treated as latent variables. Instead, the retrieved passages are explicitly concatenated with the input query and passed to the LLM as part of a single prompt. The LLM then generates the output in an auto-regressive manner, attending to the full input (query + retrieved documents) at every generation step.

Formally, the modern RAG workflow with LLMs can be described as (S. Wu et al., 2024):

$$\mathit{Retriever}(x, D) = \{z_1, z_2, ..., z_k\}, z_i \in D \tag{2.3}$$

$$p_{\text{LLM-RAG}}(y \mid x) \approx p_\theta\left(y \mid \text{concat}(x, \mathit{Retriever}(x, D))\right) \tag{2.4}$$

Where:

- x is the user query,
- D are all the retrievable documents,

- $z_1, \ldots, z_k$ are the top-k retrieved documents based on the query,
- a LLM generator p_θ, typically a transformer-based decoder nowadays (Naveed et al., 2023),
- and the input to the LLM is a structured prompt combining the query and retrieved documents.

This architecture enables the LLM to access and attend to all retrieved information simultaneously, without needing to marginalize over document-specific outputs as in the original RAG models by Lewis et al. (2020). While this method resembles RAG-Sequence structurally (one prompt per generation), it offers token-level flexibility similar to RAG-Token, thanks to the self-attention mechanism of transformer-based LLMs.

Therefore, the literature categorise the augmentation process, meaning the technical process that integrates retrieval and generation (Fan et al., 2024) (also called Fusion (S. Wu et al., 2024)), into three types: (1) Query-based RAG, (2) Logit-based RAG, and (3) Latent Representation-based RAG (P. Zhao et al., 2024). These three main types are conducted at the input, output, and intermediate layers of generators (Fan et al., 2024). As described above as the modern RAG workflow with LLMs, Query-based RAG concatenates the query and the retrieved documents based on the query as an input to the generator. In Logit-based RAG, the integration occurs during decoding by combining the generator's output logits (meaning the internal score before calculating the probability for the predicted next token) with those derived from retrieved information (S. Wu et al., 2024; P. Zhao et al., 2024). For instance, kNN-LM interpolates two next-token probability distributions during prediction, where one is generated by the language model itself and the other is derived from the nearest neighbours retrieved from all retrievable documents (Fan et al., 2024). Lastly, in Latent Representation-based RAG, the retrieved documents are encoded into intermediate representations which are fused with the generator's hidden states (S. Wu et al., 2024).

Figure 2 illustrates a LLM workflow compared to a Query-based RAG workflow. On top of the figure, the generation of a response relying only on an LLM. On the bottom of the figure, the RAG workflow is depicted, where based on the question relevant information is retrieved to enhance the context. A prompt consisting of the question and relevant context is then fed into the LLM to generate an accurate and up-to-date response (LangChain, 2024a). As an demonstration the question *"Who won the german federal election 2025?"* was fed into the LLM gpt-4o, on the top,

solely relying on the knowledge of the LLM until October 2023 and at the bottom enhanced with a Web Search able to retrieve accurate and up-to-date knowledge.

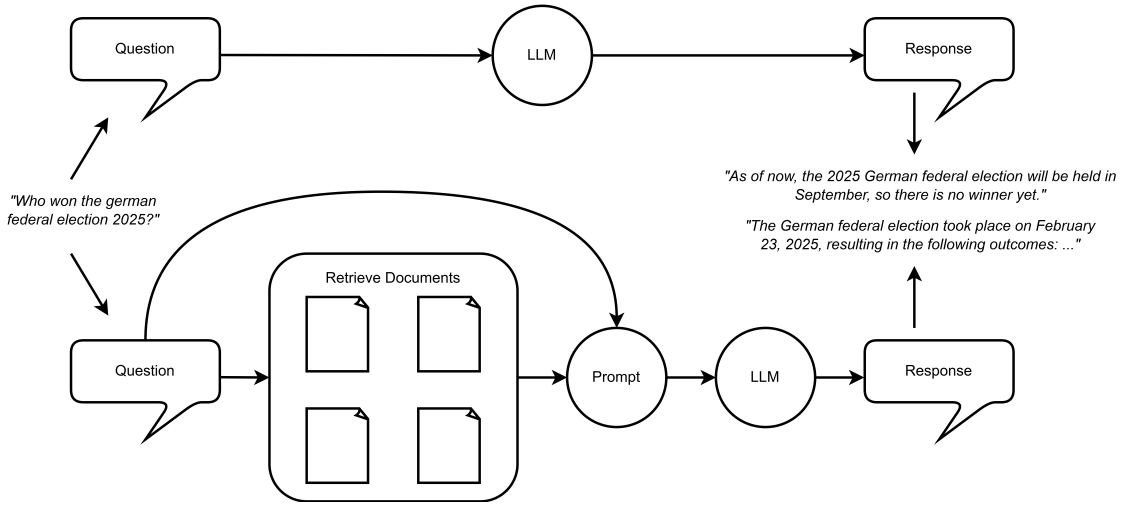


Figure 2 LLM Workflow vs. RAG Workflow (LangChain, 2024a)

2.3.1 Building a Retriever

To build a retriever for RAG it involves usually three main steps: (1) chunking the external knowledge, (2) encoding chunks, and (3) building a vector database (S. Wu et al., 2024). The process is depicted in Figure 3. The first step is chunking the external knowledge, which involves dividing large documents into semantically meaningful text chunks. The idea is that texts should be semantically independent, and contain one core idea for models to encode to prevent ambiguities (S. Wu et al., 2024). Further, long texts would result in considerable resource overheads, while shorter text chunks processed faster, accelerate the encoding process and save memory costs (S. Wu et al., 2024). Thus, the main challenge of chunking techniques is to find the best trade-off between meaningful semantics and encoding efficiency (S. Wu et al., 2024). The second step, focuses on encoding the chunks as context vectors, so called embeddings. LLM-based encoders generate dense embeddings, which capture the semantic meaning of text chunks, enabling content-based similarity search rather than simple keyword matching (S. Wu et al., 2024). These encoders leverage large-scale pre-training and powerful representation capabilities to produce rich, high-dimensional vectors (S. Wu et al., 2024). The last step is concerned with building a datastore, where data is stored as key-value pairs. On the one hand, keys are high-dimensional embeddings and on the other hand values contain the corresponding domain-specific knowledge. This type is a specialized datastore a so called vector database or vector store, which enables to efficiently store, manage, and retrieve embeddings. The vector store leverages indexing to accelerate the search process to

enable similarity search of embeddings and a given query embedding. Thus, the vector store focuses on supporting efficient Approximate Nearest Neighbour (ANN) search which enables to retrieve semantically similar chunks for a given query.

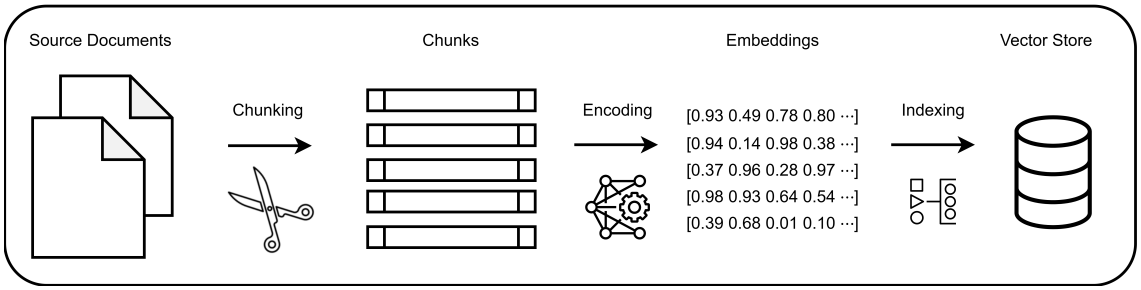


Figure 3 How to build a retriever. (Y. Wu et al., 2016)

2.3.2 Querying a Retriever

The constructed vector database now serves as the foundation for retrieving the most relevant text chunks in response to user queries. Again, S. Wu et al. (2024) describes the querying process with three mains steps: (1) encoding the query, (2) employing ANN search, and (3) post-processing. The encoding of the query uses the same encoder as the text chunks have been encoded. For example, OpenAI (2024c) provides their embedding models *text-embedding-3-small* and *text-embedding-3-large* to measure the relatedness of text strings. They have different size of dimensionality and are represented as a vector list to measure the distance between two vectors, where a small distance describes a high relatedness and a large distance suggests a low relatedness (OpenAI, 2024c). In the second step ANN search is carried out. The ANN search uses the existing indexing of the vector store to efficiently locate similar data through ANN algorithms and retrieves the associated values (S. Wu et al., 2024). Common distance functions or similarity metrics which are deployed include cosine similarity, Euclidean distance and dot product (Langchain, 2024b; OpenAI, 2024c). For example, OpenAI suggest for their embedding models cosine similarity search, as their embeddings are normalized to the length of 1, which means that it is computed faster than the dot product and it has the identical relatedness ranks when using the Euclidean distance (OpenAI, 2024c). Given a distance function to measure the similarity between the embedded query and the embedded documents, an algorithm is needed to perform an efficient search over all embedded documents to find the most similar ones (Langchain, 2024b). Therefore, S. Wu et al. (2024) presents three advanced ANN Indexing techniques for this purpose: IVFPQ (Inverted File System with Product Quantization) enables efficient and scalable approximate nearest neighbor (ANN) search by clustering data into coarse partitions and compressing

each cluster into compact sub-vectors. HNSW (Hierarchical Navigable Small World) uses a layered graph structure where high-dimensional vectors are connected to their nearest neighbours, allowing fast and accurate retrieval through probabilistic navigation. Tree-based indexing methods like KD-Trees, Ball Trees, and Annoy organize vectors into hierarchical structures using random projections to partition the space, enabling efficient ANN search through tree traversal. These ANN Indexing techniques are then leveraged to retrieve the top-k nearest neighbours of the query embedding. The post-processing step aims to refine, enhance or adapt the initial retrieval step (S. Wu et al., 2024). Common techniques include reranking of the initial retrieved documents, document repacking in the prompt or summarization of initial results to reduce redundancies and unnecessary information (X. Wang et al., 2024; S. Wu et al., 2024).

2.3.3 RAG Best Practices

Given the above insights of literature and practice a RAG System for LLMs consist of two main components: (1) a retriever, which has the external knowledge stored in embeddings and can be queried for retrieving similar documents and (2) a generation model which is able to process a prompt which is composed of the initial query and the contents of the retrieved documents. Looking into RAG literature it shows consistent efforts to enhance parts of the RAG Workflow (or also called RAG Pipeline) to improve the response of the RAG System. X. Wang et al. (2024) and P. Zhao et al. (2024) present RAG enhancements and best practices concerning the input query, retriever, generator and output response of the RAG Workflow. Due to the capabilities of LLMs not every query require RAG, thus X. Wang et al. (2024) suggests query classification based on the task prior to the retrieval step to determine if RAG is required or not. For example, translation tasks usually do not require RAG where as search questions post the training date of the LLMs require such information to be up-to-date and factually correct. Although RAG improves information accuracy and helps minimize hallucinations, repeated retrieval steps may lead to longer response times, that is why query classification should be considered when RAG is not frequently required in a System (X. Wang et al., 2024). Another technique is Query Transformation with the goal of modifying the input query in such a way to retrieve more accurate results. For example, Query2doc (L. Wang, Yang, & Wei, 2023) or HyDE (Gao et al., 2023) generate based on the initial query a pseudo document which is richer in relevant information. Instead of directly embedding the user's query and comparing it to the document embeddings, the pseudo document, which reflects a hypothetical answer to the query using a LLM in HyDE's case, is embedded and used as the basis for retrieving relevant texts from the vec-

tor database. Further, Data Augmentation is used to improve input data before retrieval, where techniques such as filtering out unnecessary content, eliminating ambiguity and vague expressions, refreshing obsolete documents, and integrating new information are leveraged.(P. Zhao et al., 2024). Furthermore, Multi Query Retriever decomposes the query into multiple clear sub-queries, which are generated by the LLM to enhance the input query for retrieval (Langchain, 2024a; P. Zhao et al., 2024). Common optimization techniques for the retriever are concerned with the chunking of the source documents. Chunking documents into smaller segments is essential to improve retrieval accuracy and to prevent issues with input length limitations in large language models (X. Wang et al., 2024). This segmentation can be performed at different levels of granularity (X. Wang et al., 2024). Token-level chunking is simple to implement but may disrupt sentence structure, potentially reducing retrieval effectiveness. Sentence-level chunking offers a practical balance by maintaining semantic coherence while remaining efficient. Semantic-level chunking, on the other hand, leverages LLMs to identify meaningful breakpoints, preserving context more effectively but at the cost of increased computational time. Thus, the chunk size plays a critical role in retrieval performance (X. Wang et al., 2024). Larger chunks offer more contextual information, which can improve understanding but also lead to longer processing times. In contrast, smaller chunks are quicker to process and tend to boost retrieval recall, though they might not contain enough context to fully capture the meaning of the original content. For example, the LLM GPT-4o has a context window of 128,000 tokens which should be considered when feeding the prompt with chunks before generation (OpenAI, 2024a). Moreover, X. Wang et al. (2024) describes that for RAG applications, choosing the right vector database directly impacts performance, especially when dealing with large-scale or cloud-based deployments. Key criteria for selecting a vector database include support for multiple index types, billion-scale vector handling, hybrid search capabilities, and cloud-native integration. Multiple indexing options allow flexibility to optimize searches based on different use cases, while billion-scale support ensures scalability for massive datasets. Hybrid search enhances accuracy by combining vector similarity with keyword-based search, and cloud-native features simplify deployment and management. (X. Wang et al., 2024) did a comparison of five open-source vector databases: Weaviate, Faiss, Chroma, Qdrant and Milvus, where Milvus stands out fulfilling all the presented criteria. Another enhancement is recursive retrieval which repeatedly queries the vector database to uncover deeper and more accurate content (P. Zhao et al., 2024). Next is the technique of re-ranking of the initial retrieved documents to improve the relevance of the retrieved documents and to ensure the most relevant information is prioritized (X. Wang et al., 2024). This phase employs more accurate and resource-demanding techniques to reorder the documents,

enhancing the similarity between the query and the top-ranked results (X. Wang et al., 2024). For example, DLM Reranking uses fine-tuned deep language models to classify document relevancy based on query-document pairs, while another technique called TILDE Reranking calculates token probabilities independently to score documents (X. Wang et al., 2024). Moreover, Document Repacking deal with the order of the retrieved documents placed into the prompt before generation with the LLM (X. Wang et al., 2024). The motivation behind it ware observed by N. F. Liu et al. (2024) who found out, that the performance of considering relevant context within the input for a LLM is the highest when it is places at the beginning or end of the prompt. Therefore, repacking the documents in an descending or ascending order based on relevancy can have an impact on the response generation (X. Wang et al., 2024). Further, the technique of Summarization is used after retrieval and before generation to check the retrieved documents if they contain redundant or unnecessary information to avoid long prompts which can slow down the generation or accuracy of the response (X. Wang et al., 2024). Lastly, prompt engineering is used for generator enhancement. In the context of RAG prompt engineering is additionally utilized to fine tune the generator to improve the quality of the final output. For example, Chain of Thought Prompt (J. Wei, Wang, et al., 2022) improve the reasoning abilities of the LLM by encouraging it to generate intermediate steps when solving complex problems. Active prompting tries to generate multiple and meaningful questions based on the user query (Diao et al., 2023). And lastly, Step-Back Prompting is a technique by asking the LLM to take a step back and look at the big picture before solving a problem, meaning an abstraction from the original query to let the LLM grasp the higher-level concepts of the question (Zheng et al., 2023). Overall, these enhancements show various ways of improving the RAG Pipeline for LLMs reflecting the relevancy and ongoing research of RAG and the striving to find a balance between the trade-off of high quality responses and resource efficient and fast runtime.

2.3.4 Evaluation of Retrieval-Augmented Generation

Evaluating RAG systems presents several unique challenges due to the complex interaction between the retrieval and generation components(Yu et al., 2024). In the survey by Yu et al. (2024), they explain that it is not enough to look at each part of the RAG pipeline separately, since the quality of the final output depends on how well the retrieved information is used in generation. This subsection summarizes the main findings of their work and gives an overview of the most important aspects of RAG evaluation. It covers the key challenges, the evaluation criteria for

both retrieval and generation, common metrics used to measure performance, and additional evaluation requirements discussed in recent literature:

A key difficulty in assessing the retrieval component lies in the dynamic and vast nature of potential knowledge sources, ranging from static databases to the constantly evolving web. This diversity demands metrics that go beyond traditional precision and recall, as they cannot fully capture the temporal relevance, contextual specificity, and filtering quality necessary in RAG applications. Moreover, information decay over time and source heterogeneity introduce additional complications for consistent evaluation. On the generation side, the main challenges include assessing the faithfulness and factual correctness of the output in relation to retrieved documents, while also ensuring relevance to the user query and overall textual coherence. Tasks such as open question answering add subjectivity, complicating the definition of a universally "correct" response (Rosset et al., 2024). Importantly, evaluating a RAG system as a whole cannot be accomplished by assessing its modules in isolation, since the quality of the generated output often hinges on how effectively retrieved content is leveraged. The combined evaluation must consider the added value of retrieval to generation, system latency, robustness to ambiguous inputs, and user performance factors such as clarity and usability.

To capture these dimensions, several key evaluation objectives have been proposed by Yu et al. (2024). For the retrieval module, two primary relations are examined: the relevance of retrieved documents to the query, and the accuracy of selected documents in relation to the total candidate documents. These are operationalized through relevance and accuracy scores. In the generation module, relevance, faithfulness , and correctness are central criteria. Relevance assesses how well the generated response aligns with the user's intent and the informational needs expressed in the question. Faithfulness measures the degree to which the generated content accurately reflects and does not contradict the retrieved source documents. Correctness evaluates whether the generated response is factually accurate when compared to a human-validated or ground truth answer. These metrics collectively measure how well the output reflects the users need, adheres to the content of the retrieved sources, and aligns with a sample response. Beyond these primary components, additional requirements for comprehensive RAG evaluation include latency, diversity, noise robustness, rejection capability when insufficient information is present, and counterfactual robustness. Further aspects such as readability, toxicity, and perplexity are also relevant when aligning chatbot behaviour with human preferences.

For retrieval quality, both non-rank and rank-based metrics are employed. Non-rank metrics include Accuracy, Precision, and Recall@k, which evaluate the relevance of selected documents without considering their order. Rank-based metrics, such as Mean Reciprocal Rank (MRR) and Mean Average Precision (MAP), account for the position of relevant results in the ranked list, rewarding systems that prioritize more useful documents earlier. For generation quality, the evaluation focuses on linguistic and semantic attributes. Classical metrics like BLEU, ROUGE, and F1 Score are still widely used to compare with reference answers. However, newer methods such as BERTScore leverage contextual embeddings to evaluate semantic similarity more robustly. Furthermore, LLMs as judges have emerged as a promising automatic evaluation technique, where large language models assess the quality of generated text across dimensions such as coherence, fluency, and informativeness, often guided by detailed scoring prompts. This shift from reference-based to context-aware evaluation allows for more nuanced, scalable, and human-aligned quality assessments.

In conclusion, the evaluation of RAG systems requires a multidimensional and integrated approach, encompassing document retrieval relevance, generative alignment and correctness, system performance metrics, and user focused considerations. While existing benchmarks address isolated aspects of this pipeline, a truly holistic evaluation framework must unify these layers to provide meaningful insights into RAG system quality and usability (Yu et al., 2024).

2.4 Economical effects of Chatbots

In recent years, various companies across industries have successfully implemented AI chatbots in customer service and support to achieve significant economic effects. For example, the fashion brand Motel Rocks integrated Zendesk Advanced AI to enhance self-service capabilities, allowing agents to focus on complex queries (Zendesk, 2024). This led to a 43% deflection of tickets, a 50% reduction in ticket volume, and a 9.44% increase in customer satisfaction. Camping World addressed its overburdened call center by deploying IBM's AI assistant Arvee, which operated 24/7 and improved customer engagement by 40%, cut wait times by 33 seconds, and increased agent efficiency by 33% (IBM, 2024). Similarly, Australian telecom giant Telstra introduced "Ask Telstra" using Microsoft's OpenAI service to summarize customer histories and fetch answers in seconds, reducing follow-up on calls by 20% and improving agent effectiveness, where 90% of agents reported enhanced performance and 84% of agents said it positively impacted customer interactions (Microsoft, 2024). ClickUp, a project management platform, adopted Maven AGI's Co-Pilot to provide AI-generated ticket summaries and suggestions, which boosted

resolution speed by 25% solves per hour and allowed the team to shift focus toward customer retention strategies (AGI, 2024). At Six Flags, a Google Cloud-powered Generative AI virtual assistant helped visitors order food, navigate the park, and access ride wait times, significantly enhancing guest experience (Flags, 2023). During the COVID-19 crisis, TEAG, a German energy supplier, implemented a chatbot in just 14 days to handle FAQs, achieving 68% automation despite limited staffing (moinAI, 2025a). Fressnapf, a pet supply retailer, achieved an 86% automation rate in customer service, with a Customer Satisfaction Score of 75% and a human takeover rate as low as 0.3% (moinAI, 2025b). Geberit, a manufacturer of sanitary products, used its chatbot to automate 47% of inquiries in multiple languages from customers and businesses and reduced support requests by 25%, while also generating leads from 10% of conversations (moinAI, 2023). Chocoversum, a chocolate museum, used a chatbot to handle 77% of inquiries and increased online ticket sales by 41%, offering services in both English and German (moinAI, 2025c). Lastly, the Frankfurter Allgemeine Zeitung (FAZ) used an AI chatbot in its subscription shop and paywall, automating 71% of inquiries and achieving an 18% click conversion rate, which led to reduced cancellation rates and improved customer service (moinAI, 2025d). These cases collectively illustrate how AI chatbots contribute to higher efficiency, reduced operational costs, improved customer satisfaction, and increased sales or conversion across both B2C and B2B settings.

3 Methodology

This thesis adopts the DSR Methodology as its overarching research framework. DSR is particularly suited for research that aims to create and evaluate innovative IT artifacts that address real-world problems. This master thesis orientates itself to the design and engineering cycle by Wieringa (2014) to present the methods used to develop the artifact and the evaluation of it. The engineering cycle consist of five phases: (1) Problem Investigation, (2) Treatment Design, (3) Treatment Validation, (4) Treatment Implementation, and (5) Implementation Evaluation. These phases are presented in the following, explaining the design problem, the plan for solving the problem, the requirements for validating the solution, and finally the procedures for the evaluation of the artifact.

3.1 Problem Investigation

The given problem context is the use case of the IT Support of the University of Muenster, which goal it is to improve the IT Support by reducing the number of repetitive requests and by that reducing personnel cost or redirecting human personal to more valuable tasks and by increasing the availability of the IT service and by making the access to information for students and staff member easier. Based on an interview with the Head of IT Services the IT Support identified a chatbot as a potential solution for this problem. So far they have tested a rule-based chatbot for this purpose but it was not able to satisfy their requirements, there were numerous reasons why such a chatbot was not deployed. The development of such a rule-based chatbot in the IT Support context required an immense effort and constant maintenance to program the rules by hand for every possible question in that context, thus having a limited flexibility and scalability. Further, the rule-based chatbot was not suitable for open and complex questions which is common for support requests. Lastly, outsourcing was an option, but it was too costly for the limited value offered by the rule-based chatbot and would have entailed a significant loss of control over system customization and maintenance. As a result of these factors, the project was ultimately suspended. Now with the rise of RAG for LLMs a new solution, in terms of DSR a treatment of a real-world problem, emerged for developing a domain-specific chatbot for the IT Support. Thus, this master thesis explores the artifact of a RAG-based chatbot for the IT Support of the University of Münster. According to Wieringa (2014) design problem template this design problem could be formulated as follows: **Improve** the IT Support Service **by designing** a RAG-based chatbot **that satisfies** accuracy, availability, flexibility and scalability requirements **so that** the IT Support personnel is no longer occupied with a high amount of repetitive

requests and so that students and staff members can access the 1st-Level IT Support at any time and get factual correct answers faster and more conveniently.

3.2 Treatment Design

The artifact developed in this master thesis is a RAG-based chatbot designed to improve access to IT support information at the University of Münster. To effectively address the problem context and fulfil stakeholder needs, the chatbot must satisfy a set of design objectives derived from the identified requirements.

3.2.1 Design Objectives

Based on the analysis of the problem context and stakeholder expectations, the following design objectives were formulated:

- **O1:** Enable easy and web-based access to the chatbot.
- **O2:** Ensure accurate information retrieval by loading the original sources of the CIT website.
- **O3:** Ensure a scalable deployment capable of handling thousands of simultaneous users.
- **O4:** Use a modular software architecture that supports the integration of both internal and external LLMs.
- **O5:** Maintain conversational context through persistent chat history.
- **O6:** Allow future extensibility for additional data sources and potential chatbot features.

3.2.2 System Architecture and Implementation Design

To meet these objectives, a modular web application was designed, consisting of the following core components:

- **Frontend:** A user-friendly web interface with a chat component, allowing users to interact with the chatbot in a natural language format. This frontend can either be integrated into the official CIT website or hosted separately with a shared access link.

- **Backend:** A RAG-based retrieval logic is implemented in the backend. It uses a combination of a vector store and a generation module to produce contextually relevant and accurate responses. All relevant documents from the CIT website are scraped using automated scripts then embedded and lastly stored in a vector database.
- **LLM Integration:** The generation component is model-agnostic and supports both own hosted and external LLMs. This allows flexibility in switching or upgrading models in the future as new capabilities become available.
- **Source Attribution:** All responses include source links to the relevant CIT webpages, enabling users to verify and explore information further.
- **Memory Component:** To enhance the quality of the interaction, the chatbot maintains a memory of the chat history, ensuring coherent and context-aware responses across multi-turn dialogues.

3.2.3 Deployment Considerations

Given that the University of Münster serves over 42,000 students and 7,000 staff members, the system is designed for high scalability and availability. The application is deployed in a containerized environment on the university's Kubernetes cluster to ensure efficient orchestration and fault tolerance.

For the initial release, the chatbot was deployed in a controlled environment with exclusive access granted to a limited group of test users. This approach allows for a thorough evaluation of the system before wider public deployment. Evaluation results are discussed in Chapter 5.

With the above design and architectural decisions, a functional RAG-based chatbot was developed. Implementation details are described in Chapter 4.

3.3 Treatment Validation

Treatment validation involves predicting the outcomes of applying the artifact within the intended problem context. As shown in Chapter 2.2.5, RAG-based chatbots have demonstrated their utility across diverse domains beyond academia. This suggests strong potential for domain-specific applications, including university IT support. Additionally, Chapter 2.4 presents evidence from similar artifacts that have improved service efficiency and yielded positive economic effects.

3.3.1 Expected Effects for Users

The deployment of a RAG-based chatbot by the IT support of the University of Münster is anticipated to result in the following user benefits:

- **Instant access to support:** Users receive accurate answers immediately without having to wait in hotline queues or for email replies.
- **Availability outside working hours:** The chatbot is accessible 24/7, which is especially helpful for students and staff working during evenings or weekends.
- **Reduced frustration:** Users are spared from navigating lengthy documentation and can ask their question directly.
- **Context awareness:** The chatbot can adapt responses based on user status (e.g. student or staff) or technical parameters (e.g. operating system).
- **Multilingual support:** International students and staff benefit from communication in their preferred language.
- **Potential barriers:** Some users may hesitate to adopt the chatbot, particularly for sensitive or complex issues where human support is preferred.

Overall, increased user satisfaction is expected due to reduced search time and the convenience of conversational interaction with the chatbot.

3.3.2 Expected Effects for IT Support Staff

For the IT support team, the chatbot is expected to deliver operational relief and efficiency:

- **Reduced workload:** Frequently asked questions (e.g., "How do I connect to the VPN?") can be answered automatically.
- **Lower ticket volume:** Fewer repetitive questions mean more time for complex, high-priority cases.
- **Improved resource allocation:** Staff can focus on tasks requiring deeper technical insight.
- **Fewer interruptions:** Less phone calls leads to less interruptions which leads to more focused work.

- **Support for onboarding:** New staff members can use the chatbot as a knowledge base to quickly access internal resources and guides.
- **Increased job satisfaction:** Less time spent on routine replies allows staff to concentrate on meaningful problem-solving.

3.3.3 Expected Economic Effects for the University

For the institution, the following economic benefits are anticipated:

- **Reduced support costs:** Automation of routine questions decreases the need for human intervention in 1st-Level Support.
- **Lower operational overhead:** The chatbot as a new service offer reduces the reliance on phone and email channels.
- **Scalable support:** The chatbot can handle high demand (e.g. at semester starts) without requiring additional staff.
- **Long-term savings:** Cost reductions through delay or elimination of hiring additional support staff.

3.4 Treatment Implementation

The chatbot was deployed productively via the internet. However, access was initially limited to a selected group of users, ensuring that feedback could be collected in a controlled environment before public release. This phased rollout helped assess technical performance, user satisfaction, and usability while minimizing risk.

The chatbot is hosted on the university's Kubernetes cluster and built using the LangChain framework, with a Streamlit front-end. Relevant content from the CIT website was scraped using BeautifulSoup and stored in a vector database (ChromaDB). The system supports integration with both internal and external LLMs, allowing future upgrades with more advanced models. Source links are included in the responses for user transparency and traceability. Details on the implementation and design decision are presented in chapter 4.

3.5 Implementation Evaluation

The evaluation of the implemented chatbot follows a mixed-methods approach, combining both quantitative and qualitative data collection techniques to comprehen-

sively assess the system's performance and stakeholder impact. This includes three surveys targeting different evaluation objectives:

- **Survey 1 – Users:** This survey captures user preferences regarding information retrieval strategies (e.g., Google, CIT website, generic LLM, or the RAG-based chatbot) and the perceived usefulness and convenience of the chatbot.
- **Survey 2 – IT Staff:** This survey investigates the chatbot's impact on internal work processes, including the reduction of repetitive questions, perceived workload changes, and the usefulness of the chatbot as a support tool.
- **Survey 3 – Response Accuracy:** This evaluation assesses the factual correctness of chatbot-generated answers for the top 8 most frequently asked support questions. These questions were provided by the Head of IT Services. Each generated answer was manually compared to the official information published on the CIT website to verify whether the response accurately reflects the content of the source. For that a binary scoring system is applied across four key dimensions: factual correctness, completeness, clarity, and correct source attribution. Yielding an accuracy score from 0 to 4 per response. The final accuracy score is derived by averaging the results across all tested questions.

Survey 1 and 2 incorporate both structured Likert-scale items (1–5) for statistical analysis and open questions to gather qualitative insights, such as user experiences, feedback, and suggestions for improvement. To analyse the responses for the open questions, a qualitative content analysis was conducted. The responses were systematically examined to identify recurring themes and develop relevant categories. These categories enabled the summarization of feedback across multiple participants and supported the identification of key issues and user perceptions related to the chatbot. To support the qualitative content analysis, the LLM ChatGPT (GPT-4 by OpenAI) was employed to serve as a classification engine. The tool was primarily used to assist in the initial structuring of responses, identifying recurring themes, and formulating preliminary categorizations. Its use served as a means to enhance efficiency during the coding process. All outputs generated by the AI model were critically reviewed and cross-checked against the original responses. Modifications and corrections were made where necessary to ensure the accuracy and validity of the final analysis. Thus, while ChatGPT provided useful support in processing and organizing the qualitative data, the final categorization, coding, and interpretation

were carried out and validated by the author in line with established principles of qualitative content analysis.

Survey 3 provides an accuracy evaluation framework comparing the chatbots accuracy in responding to the top 8 frequently asked questions. A detailed analysis and the results of the evaluation are presented in Chapter 5.

3.6 Limitations

Despite the structured evaluation approach, several limitations must be acknowledged. The evaluation was conducted in a controlled setting and has not yet been tested in real operational support workflows. The number of participants was limited to 25 end users and 10 IT support staff members, which restricts the generalizability of the results and the statistical power of the findings. Additionally, long-term user behaviour, chatbot fatigue, or evolving needs over time were not captured in the short evaluation window.

Another methodological limitation concerns the insufficient exploration of stakeholder requirements prior to the implementation phase. The initial needs were only discussed in an interview with the Head of IT Services, where the focus primarily lay on identifying a feasible technical solution rather than thoroughly investigating the expectations of end users or support staff regarding user experience and functional features. Although a rule-based chatbot had previously been evaluated and dismissed due to high maintenance efforts, lack of flexibility, and poor scalability for open questions, no extensive follow-up study or survey was conducted to define what a successful alternative chatbot should deliver from a user-centric perspective for this use case. As a result, the development phase may have missed the opportunity to incorporate more tailored requirements related to usability, conversational tone, personalization features, or accessibility. Conducting a more comprehensive needs assessment beforehand could have provided a clearer design focus and ensured a higher alignment with actual user and staff expectations.

From a technical perspective, the implementation is based on the LangChain framework, which was selected due to its rapid development capabilities and extensive support for RAG workflows. However, no comparison was made with alternative frameworks such as Haystack or LlamaIndex, and as such, it remains unclear whether LangChain offered the most efficient or reliable solution for this use case. Its reliance on internal abstractions and predefined functions may also have limited the opportunity for deeper customization or optimization.

Another limitation of the current implementation is the absence of query classification and reranking techniques within the RAG-logic. While the system was designed to always provide the source of the most relevant document out of the top 5 results, the lack of query classification means that a source was provided for every question, even if it was not contextually appropriate. Additionally, without reranking or other advanced techniques, the document ranking was solely determined by the similarity search relevance order in the ChromaDB vector store. As a result, the top document was not necessarily the most relevant or the most useful to the user, potentially leading to less accurate or less helpful responses in some cases. This limitation opens the door for future exploration of additional RAG enhancements.

Additionally, the application was designed to be model-agnostic, and during development, different large language models were tested. This included local models provided by the University of Münster. While the Llama-3.3-70B model performed reasonably well, the newer mistral-small model exhibited unstable and unexpected behaviour during internal testing. These models were only evaluated in development and not exposed to user-facing deployment due to accuracy issues. Ultimately, gpt-4o-mini was chosen as the most reliable option for delivering consistent and high-quality responses. However, gpt-4o-mini is an external service, which introduces potential privacy concerns, especially when handling user inputs that may contain sensitive information. Although the CIT website content is publicly accessible and does not inherently pose a confidentiality risk, this limitation must be carefully considered when planning a broader deployment of the application, particularly in regard to data protection for end users and compliance with university IT data privacy policy.

4 Implementation of the RAG-based Chatbot

This chapter details the technical implementation of the RAG-based chatbot developed as part of this thesis. It begins with an overview of the overall application architecture, followed by a discussion of the selected frameworks and tools, including the rationale behind their use. Subsequently, the application logic is described in detail, encompassing the preprocessing of documents, the individual components of the RAG workflow, session management, and the integration with the frontend interface. Finally, the chapter concludes with a description of the deployment process on the Kubernetes cluster of the University of Münster.

For comprehensive insight into the source code, a GitLab repository was created and made publicly accessible via the following URL:

https://zivgitlab.uni-muenster.de/j_alth12/rag-master-thesis.

The deployed instance of the RAG-based chatbot can be accessed at:

https://rag-thesis.uni-muenster.de/.

4.1 Architectural Design

The developed application is implemented in the Python programming language using the Visual Studio Code IDE. The system architecture follows a modular design, with the LangChain framework serving as the core for implementing the RAG workflow. For content extraction, BeautifulSoup is used to scrape relevant information from the official CIT website. These documents are then embedded using OpenAI's embedding models, and the resulting vector representations are stored in a ChromaDB vector store. The language generation is powered by the gpt-4o-mini model, which was selected due to its strong performance during internal testing. The user interface is built with Streamlit, which also provides session state capabilities to maintain chat history during user interactions. Finally, the application is deployed on the Kubernetes cluster of the University of Münster. For this purpose, a Docker image was created and published via Docker Hub, allowing containerized deployment within the cluster environment.

Figure 4 provides a visual overview of the system architecture, illustrating the core components, including the programming language, frameworks, backend logic, and frontend technologies. The following subsections provide a detailed description of each framework and tool used in the implementation.

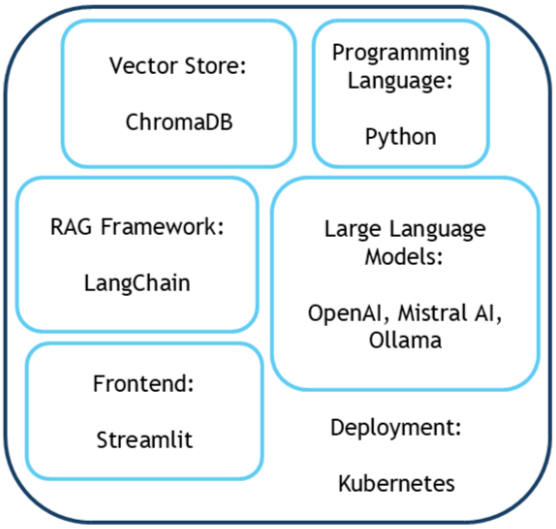


Figure 4 Application Frameworks and Tools

4.1.1 LangChain

LangChain is a comprehensive framework designed to facilitate the development of applications powered by LLMs. In this project, LangChain version 0.2 (LangChain, 2024c) was utilized to implement the RAG logic. At its core, LangChain provides a modular and extensible system architecture that supports chaining together various components such as prompt templates, retrievers, LLMs, output parsers, and vector stores. Central to its framework is the LangChain Expression Language (LCEL), a declarative syntax that enables developers to build chains, from simple prompts feeding into LLMs to complex multi-step workflows with production readiness in mind. LCEL supports synchronous and asynchronous execution, nearly real-time streaming output, parallel execution, retries, fallbacks, and intermediate result access (LangChain, 2024d). These capabilities significantly reduce latency (e.g., time-to-first-token) and enhance robustness under concurrent loads (LangChain, 2024d). LangChain introduces a unified "Runnable" interface, enabling consistent invocation across components with methods such as invoke, stream, and their asynchronous counterparts like ainvoke or astream. Additionally, LangChain offers extensive support for handling various message types with role-based metadata, a robust document processing pipeline (including loaders, transformers, and splitters), and native integration with multiple vector stores and embedding providers. This modularity allows for precise control over document ingestion, transformation, storage, and retrieval, which is critical for the design of RAG-based systems. Chat history management and output parsers further strengthen its utility in dialogue-centric applications by maintaining conversational context and structuring model outputs.

One of the main reasons for selecting LangChain for the development of the RAG-based chatbot was its rich ecosystem, comprehensive documentation, and extensive support for third-party integrations, including leading LLM and vector store providers. Moreover, LangChain's tight coupling with evaluation tools such as LangSmith facilitates rigorous monitoring and iterative refinement in production settings. The learning curve was eased considerably by well structured tutorials, particularly for RAG implementations. Notably, the framework's latest version (v0.3) marks a paradigmatic shift from static chains to agent-based architectures. Agents are systems that levering LLMs as reasoning engines capable of dynamically invoking tools and taking contextual actions in a more adaptive and interactive way (LangChain, 2024b). This trajectory aligns with the broader movement in LLM development toward autonomous and decision-capable agents, making LangChain not only a practical but also a forward-looking choice for modern LLM application development.

4.1.2 ChromaDB

ChromaDB is an open-source vector database designed for AI applications. It comes with powerful built-in capabilities such as embedding support, vector search, document storage, full-text search, metadata filtering, and multi-modal retrieval—all in a single package (Chroma, 2024). It supports both local persistence, ideal for managing private or self-hosted documents, and cloud-based management via Chroma Cloud. Hybrid search is also supported (X. Wang et al., 2024). ChromaDB integrates seamlessly with LangChain for creating and managing vector stores, making it a great fit for RAG applications. ChromaDB was choosen for this project because of its simple setup, native integration with LangChain, and suitability for small to medium-scale use cases. Since the documents used are homogeneous in structure and under 1000 in number, ChromaDB's performance and scalability are more than sufficient for current and foreseeable requirements.

4.1.3 OpenAI GPT-4 series and Embedding model

For the implementation of the RAG-based chatbot, the gpt-4o-mini model from OpenAI's GPT-4 series was selected. This LLM represents a powerful, state-of-the-art system that excels in reasoning, problem-solving, and maintaining contextual coherence (OpenAI, 2025). During development and testing, gpt-4o-mini demonstrated itself to be especially robust and consistent, both in terms of response quality and stability across diverse user inputs. Moreover, the pay-as-you-go pricing model offers flexibility and cost-efficiency, making it particularly attractive for academic and

prototype use cases. Gpt-4o-mini also integrates well into RAG architectures due to its strong instruction-following capabilities and ability to effectively ground answers in retrieved context (J. Zhou et al., 2023).

To complement the language model, OpenAI's text-embedding-3-large was used as the embedding model. This model transforms text into high-dimensional vector representations and supports customization of the embedding dimension, allowing trade-offs between computational cost and representational richness (OpenAI, 2024c). The ability to reduce vector size while preserving semantic relationships is particularly useful for efficient storage and fast similarity search in vector databases (OpenAI, 2024c). Together, with gpt-4o-mini strength in instruction following and OpenAI's text-embedding-3-large model which is highly tuned for semantic accuracy it enables a effective pair in a RAG system (OpenAI, 2024b). During the development and testing of the RAG-based chatbot, testing other embedding models to improve accuracy was not prioritized, as the current setup performed well from the beginning. Moreover, the application is model-agnostic for the LLM, allowing for substitution with other models. However, both embedding model and LLM selection can be further explored on RAG system performance.

4.1.4 BeatifulSoup

BeautifulSoup is a Python library used for parsing HTML and XML documents, offering an intuitive interface for navigating, searching, and modifying page content (Richardson, 2004–2025). It is particularly effective for extracting specific information from web pages when an official API is unavailable or insufficient (Richardson, 2004–2025). BeautifulSoup supports multiple parsers and allows developers to target elements using tag names, attributes, and CSS selectors, making it well-suited for structured content extraction (Richardson, 2004–2025). In this project, BeautifulSoup was used to scrape relevant content from the official website of the CIT at the University of Münster. The scraped data included all pages of the CIT webpage which are reachable from its site map, which were subsequently cleaned and preprocessed for embedding into the retrieval pipeline. By automating the data collection process, BeautifulSoup enabled consistent and up-to-date access to the knowledge base, forming the foundation for accurate and contextually relevant chatbot responses.

4.1.5 Streamlit

Streamlit is an open-source Python framework designed to facilitate the rapid development of interactive web applications for data science and machine learning use cases (Snowflake Inc., 2025e). It allows developers to create rich, responsive user interfaces with minimal frontend expertise by writing Python scripts. Its declarative syntax and automatic UI rendering make it particularly suitable for prototyping and deploying AI-powered applications, including those involving large language models (LLMs) and chatbot systems. One of Streamlit's key advantages is its seamless integration with the Python ecosystem, enabling end-to-end development without context switching between languages or frameworks. For chatbot applications, it provides robust support for managing conversational flow and user input through session states, which makes it possible to preserve chat history and intermediate outputs across user interactions (Snowflake Inc., 2025f). This feature proved to be useful for debugging and testing during the development phase, particularly in visualizing the scraped documents, retrieved documents, and final response stream with metadata enrichment. Furthermore, Streamlit supports function caching, which significantly improves runtime performance by avoiding unnecessary recomputation which is especially beneficial in applications where data updates or vector store indexing occur infrequently, as every interaction with the webpage runs the script in the background again (Snowflake Inc., 2025d). Its extensive API documentation and active community ensure smooth onboarding and quick problem-solving. Additionally, Streamlit offers clear guidance on containerization using Docker and supports deployment via Kubernetes, making it an excellent choice for scalable and maintainable frontend deployment in production environments (Snowflake Inc., 2025a, 2025b). Due to these characteristics, Streamlit was selected as the frontend framework for the chatbot application, offering an ideal balance between development speed, functionality, and production readiness.

4.2 Application Logic

This section describes step-by-step how the application runs and goes into implementation details. Again the full source code is available under the URL: https://zivgitlab.uni-muenster.de/j_alth12/rag-master-thesis.

4.2.1 Source Document Preparation

The preparation of source documents begins with scraping web content from the CIT website, which contains comprehensive information about the university's IT

services. The scraping process starts from the CIT site map and recursively collects all links that match the CIT base URL pattern (https://www.uni-muenster.de/IT/) up to a depth of five levels. The recursion halts once no further valid links under the base URL are found or is currently set to seven levels deep. To maintain up-to-date content, scraping is scheduled once per day. This is controlled by a function named *should_update()*, which compares the timestamp of the last update with the current date to determine whether a new scraping cycle should be initiated. This automation ensures that newly added or modified documents are continuously integrated without manual effort. Additionally, the scraper compares existing and new links to avoid overwriting previously stored data in the event of temporary unavailability or removal of pages. All collected links are persisted in a JSON file for further processing. Following the persistence of relevant URLs, the web content is extracted and stored using the function *get_persistent_docs()*. This function utilizes a customized *WebBaseLoader* in combination with BeautifulSoup. To focus exclusively on relevant content, BeautifulSoup's *SoupStrainer* is used to restrict parsing to `div` elements with the class *module-content*, which contain the main body of information. This approach excludes irrelevant components such as navigation sidebars. It is important to note that this approach relies on the specific structure of the underlying HTML files and may require adjustments if applied to other websites in the future, as HTML structures can vary significantly. The LangChain *WebBaseLoader* returns the content as a list of *Document* objects, each containing metadata (e.g. the source URL) and page content in plain text. However, this format does not retain hyperlinks, which are important for the chatbot's ability to provide source references. To overcome this limitation, a custom class *LinkPreservingWebLoader(WebBaseLoader)* was developed to preserve hyperlinks by appending the target URL in parentheses after the anchor text (e.g. "Click here" (Link)). This enhancement enables the chatbot to include functional links within its responses. The resulting list of *Document* objects is persisted in a `.pkl` file and serves as the basis for the embedding and storage process. The `.pkl` format was chosen to store the list of documents because it allows efficient serialization and deserialization of Python objects, as pickle is binary, preserving their structure and content for seamless reuse without the need for reprocessing (IONOS, 2024).

4.2.2 Embedding and Vector Store Initialization

The source code of the embedding model, vector store, and retriever is shown in Listing 1. The next step involved building a retriever using LangChain's vector store utilities. For this purpose, the Chroma class was employed to initialize a Chroma vector store instance, which persists data locally and utilizes OpenAI's embedding

model *text-embedding-3-large*. To reduce computational costs and memory usage during embedding and retrieval, the dimensionality of the generated embeddings was reduced from the original 3072 dimensions to 512 by using the dimensions parameter provided by the OpenAI API (OpenAI, 2024c). This dimensionality reduction results in a significantly smaller vector representation while preserving the semantic properties of the embeddings (OpenAI, 2024c). Initial testing with the shortened 512-dimensional vectors revealed no substantial loss in retrieval performance, making it a practical configuration for this application. Nonetheless, the relationship between embedding dimension, retrieval quality, and resource consumption presents an interesting avenue for further research, especially in the context of large scale or domain-diverse datasets. During development, a notable challenge emerged when attempting to embed the full corpus of documents in a single request. This approach exceeded the maximum token limit allowed per request by the OpenAI embedding API, resulting in the following error:

```
openai.BadRequestError: Error code: 400 - {
    'error': {
        'message': 'Requested 600275 tokens, max 600000 tokens per
           request',
        'type': 'max_tokens_per_request',
        'param': None,
        'code': 'max_tokens_per_request'
    }
}
```

This constraint necessitated batching the embedding process to comply with the API's maximum token limit of 600,000 per request. To address the token limit imposed by the OpenAI embedding API, a batching logic was implemented. This was achieved by utilizing OpenAI's Tokenizer to compute the token count for each document (OpenAI, 2024d). A custom function, *batch_documents_by_tokens(documents)*, was developed to divide the input documents into batches such that each batch remained within the 600,000-token limit, returning a list of batches. These batches were then processed sequentially, embedding and adding them to the vector store without exceeding the token constraints of a single API request. As a result, a fully populated vector store containing all embedded documents was successfully constructed. To ensure that the vector store reflects the most recent data, the batching function was designed with caching functionality. Since the document import occurs daily, the previous vector store instance, if existent, is deleted and recreated using the current document set. This guarantees that modifications to existing documents or additions of new documents are consistently reflected in the embeddings stored. Once populated, the vector store can be queried to retrieve semantically relevant documents. For this, a function named *get_retriever()* was implemented. It performs a similarity search between the embedding of a user query and the stored

document embeddings, returning the top five most relevant results. The underlying similarity metric is determined by the vector store implementation and the scale of the embeddings (LangChain, 2023b). Given that OpenAI's embedding vectors are normalized to length 1, cosine similarity and Euclidean distance are monotonically related, thereby producing identical ranking outcomes. Formally this can be shown, derived from Terveer (2019):

Let the vectors $\mathbf{a} = (x_1, x_2)$ and $\mathbf{b} = (y_1, y_2)$ have unit length, i.e.:

$$\|\mathbf{a}\| = \|\mathbf{b}\| = 1.$$

The cosine of the angle φ between the two vectors is given by the dot product formula:

$$\cos(\varphi) = \mathbf{a} \cdot \mathbf{b} = x_1 y_1 + x_2 y_2.$$

And the Euclidean distance between the vectors $\mathbf{a}$ and $\mathbf{b}$ is defined as:

$$\|a - b\|^2 = (x_1 - y_1)^2 + (x_2 - y_2)^2$$

Using $\|a\|^2 = 1$ and $\|b\|^2 = 1$, we get:

$$\|a - b\|^2 = 2 - 2(x_1 y_1 + x_2 y_2)$$

Substitute $\cos(\varphi) = x_1 y_1 + x_2 y_2$:

$$\|a - b\|^2 = 2 - 2\cos(\varphi)$$

Finally, taking the square root of both sides:

$$\|a - b\| = \sqrt{2(1 - \cos(\varphi))}$$

Here we can see that cosine similarity and the Euclidean distance are monotonically related because if the cosine similarity increases the Euclidean distance decreases and vice versa. Therefore, ranking vectors by cosine similarity or by Euclidean distance

gives the same ordering of similarity when the vectors are normalized to length 1. LangChain also supports more search types such as maximal marginal relevance based on a similarity score threshold. These possibilities also opens up an area for further research to measure retrieval quality based on search types which was not in the scope of this work. LangChain additionally supports advanced search methods such as Maximal Marginal Relevance, which combines similarity scoring with a diversity criterion or similarity score thresholds to retrieve relevant documents independent of the k value, which reflects the number of documents to retrieve (LangChain, 2025a, 2025b). These extended capabilities offer promising avenues for future research, particularly with respect to evaluating retrieval quality across different search strategies but this is an aspect that lies beyond the scope of this work.

```
def get_embeddings():
    return OpenAIEmbeddings(
    model="text-embedding-3-large",
    dimensions=512
)

def vector_store():
    """
    Creates and caches a Chroma vector store while handling token batching to stay within API limits.

    - Deletes and recreates the vector store if it already exists.
    - Retrieves documents and adds them in batches.
    - Ensures that each batch remains below the OpenAI token limit.
    - Persists the vector store after processing all documents.

    :return: A Chroma vector store containing all processed documents.
    """
    db_path = "./db"  # Directory where the Chroma vector store will be saved

    # Step 1: Check if the vector store already exists and delete it if present
    if os.path.exists(db_path):
        shutil.rmtree(db_path)  # Remove the existing vector store
        logging.info("Previous vector store has been deleted.")

    # Step 2: Create a new Chroma vector database
```

```
26      vectordb = Chroma(persist_directory=db_path,
            embedding_function=get_embeddings())
27
28      # Step 3: Fetch the latest documents from persistent
            storage
29      documents = get_persistent_docs()
30
31      # Step 4: Process documents in batches to ensure they
            stay below the OpenAI token limit
32      for batch in batch_documents_by_tokens(documents):
33          vectordb.add_documents(batch)
34
35      # Step 5: Persist the vector store after processing all
            the documents
36      vectordb.persist()
37      logging.info("New vector store successfully created and
            saved.")
38
39      # Step 6: Return the created vector store
40      return vectordb
41
42
43  def get_retriever():
44      """
45      Retrieves a similarity-based retriever from the Chroma
            vector store.
46
47      This function returns a retriever that allows searching
            for the top-k most
48      similar documents based on their vector embeddings.
49
50      :return: A retriever object that can be used to fetch
            relevant documents.
51      """
52      return vector_store().as_retriever(
53          search_type="similarity",  # Use similarity-based
                retrieval
54          search_kwargs={'k': 5}     # Retrieve the 5 most
                relevant documents
55      )
```

Listing 1 Defining the Embedding Function, Chroma Vector Store and Retriever

4.2.3 RAG Logic

The query-based RAG pipeline implemented in this work consists of three primary steps. The first step involves query transformation, where the user's current input is reformulated based on the preceding chat history. This transformation ensures that the generated query reflects the full conversational context before the retrieval

of relevant documents in the subsequent step. For this purpose, LangChain provides a pre-built utility function: *create_history_aware_retriever(llm: LanguageModelLike, retriever: RetrieverLike, prompt: BasePromptTemplate)*, which accepts a Large Language Model (LLM), a retriever, and a prompt for query contextualization as input. It returns a list of documents that are most relevant to the reformulated query (LangChain, 2023a).In this setup, OpenAI's GPT-4o-mini model is used as the LLM, while the retriever corresponds to the previously instantiated ChromaDB instance. The prompt employed for query contextualization is derived from LangChain's template library and is structured as a system message. This message provides explicit instructions to the LLM to rephrase the user input into a standalone question that incorporates prior conversational context. The resulting standalone query is then embedded and passed to the retriever to perform a similarity search, retrieving the top five documents most relevant to the transformed query. The complete implementation of the contextualization logic is provided in Listing 2.

```
def history_aware_retriever_with_contextualization():
    """
    Creates a history-aware retriever that can formulate standalone questions
    from a chat history and the latest user input. The history-aware retriever
    ensures that the context of previous interactions is taken into account
    when processing user queries.

    - Uses a system prompt to contextualize the user's question.
    - Combines the chat history with the latest user input to generate a standalone question.
    - Returns a retriever that integrates contextualization and history-aware search.

    :return: A history-aware retriever ready for use in a conversational system.
    """

    # Step 1: Define the system prompt that provides instructions for contextualizing the question
    contextualize_q_system_prompt = (
        "Given a chat history and the latest user question "
        "which might reference context in the chat history, "
        "formulate a standalone question which can be understood "
        "without the chat history. Do NOT answer the question, "
```

```
        "just reformulate it if needed and otherwise return it as is."
    )

    # Step 2: Create a ChatPromptTemplate with placeholders for the chat history and the latest user input
    contextualize_q_prompt = ChatPromptTemplate.from_messages(
        [
            ("system", contextualize_q_system_prompt),  # System message with instructions
            MessagesPlaceholder("chat_history"),        # Placeholder for chat history
            ("human", "{input}"),                       # Placeholder for the user's latest input
        ]
    )

    # Step 3: Create a history-aware retriever by combining the contextualization prompt and retriever
    history_aware_retriever = create_history_aware_retriever(
        get_lmm(),          # Language model that will be used for contextualization
        get_retriever(),    # Base retriever which does a similarity search
        contextualize_q_prompt  # The prompt that helps to generate a standalone question
    )

    # Return the created history-aware retriever
    return history_aware_retriever
```

Listing 2 Defining a History-Aware Retriever with Contextualization

The third step involves combining the reformulated query, the retrieved contextual documents, and a predefined system prompt, which together form the input for the LLM. For this purpose, a dedicated function named *rag_chain()* was implemented. This function returns an LCEL Runnable object that can be invoked to produce a dictionary containing both the context and the generated answer in response to a user query. Initially, the function uses *create_stuff_documents_chain(llm, qa_prompt)* to construct a chain that integrates the relevant context, the conversation history, and the current user input into a unified prompt for response generation. This chain is subsequently embedded into a final RAG pipeline using the LangChain utility function *create_retrieval_chain(retriever, question_answer_chain)*. This function combines the output of the history-aware retriever, meaning the retrieved documents, with the final generation chain, enabling the LLM to produce contextually grounded responses. The system prompt used within the chain is specifically

crafted to instruct the language model to act as an IT support assistant for the University of Münster. It imposes several behavioural constraints: the assistant should only respond to IT-related inquiries within the university's context, refer users to the official CIT website when additional information is required, and escalate complex or unresolvable questions to the university's IT hotline. The assistant is also instructed to answer in the same language used by the user and to follow detailed procedural rules when providing installation or troubleshooting support. This includes delivering step-by-step instructions, incorporating hyperlinks where relevant, and prompting for clarifying information such as the operating system in use. Moreover, the assistant is designed to exhibit an approachable and empathetic tone by using emojis and engaging in intelligent follow-up questions to refine its understanding of the user's needs. The full content of the system prompt, as developed through a trial-and-error process, is shown in Listing 3. This stage of the pipeline underscores the significance of prompt engineering in achieving desired conversational behaviour and generation quality. Optimizing the prompt remains an open area for further experimentation, where techniques such as document repacking or advanced prompt restructuring may further enhance performance (N. F. Liu et al., 2024; X. Wang et al., 2024). By invoking the *rag_chain()* function, the full RAG workflow is activated, starting from the transformation of the user query, continuing with retrieval of relevant documents, and concluding with context-aware response generation, thereby enabling the chatbot to provide coherent and domain-specific answers to user questions.

```
1  def rag_chain():
2      """
3      Creates an IT support assistant chain for the University
           of M nster that answers IT-related queries.
4
5      This function defines a system prompt for the assistant,
           creates a question-answer chain with context,
6      and integrates a history-aware retriever for
           contextualized querying.
7
8      - The system prompt defines the behavior, rules, and
           expectations of the IT support assistant.
9      - The assistant uses a ChatPromptTemplate to incorporate
           chat history and the latest user input.
10     - The history-aware retriever is used to ensure the
           assistant can handle and contextualize ongoing
           conversations.
11
12     :return: A chain of the IT support assistant capable of
           answering IT-related questions.
13     """
14
```

```
15      # Step 1: Define the system prompt that dictates the
            behavior of the IT support assistant
16      system_prompt = (
17          """
18          You are an IT support assistant for the University of
                M nster.
19          Your sole purpose is to assist with IT-related issues
                that students and employees of the university may
                encounter.
20
21          IMPORTANT RULES:
22          - Focus on answering IT-related questions concerning
                the university's IT services.
23          - Also try to answer all questions concerning the
                University of M nster.
24          - Try to predict if they mean something else or
                similar with the question.
25          - If you do not know the answer
26              1. Ask smart counter questions which can help to
                    clarify the question.
27              If the question is clear and you still cannot
                    answer.
28              2. Refer to the CIT Website: https://www.uni-
                    muenster.de/IT/services/index.html
29              Lastly, if you still do not know and the question
                     is super complex then
30              3. refer to the right Contact Persons(
                    Ansprechpartner) or at least the IT Hotline (
                    https://www.uni-muenster.de/IT/ansprechpartner
                    /index.html#id0) for general questions.
31          - Always answer in the same language as the user.
32
33
34          INSTALLATION & TROUBLESHOOTING:
35          - For installation guides:
36            Always ask for the operating system before
                  providing steps.
37            List **all required prerequisites** before
                  explaining the installation.
38            Provide a **detailed step-by-step** guide with
                  numbered steps.
39            Do not skip any important configurations or
                  settings.
40            Be as detailed as possible list **everything**!
41            Always provide Hyperlinks when they are part of the
                   guide.
42          - If troubleshooting an issue:
43            Ask clarifying questions to understand the exact
                  problem.
44            Provide **structured debugging steps**, from basic
                  to advanced.
```

```
45
46          AUTHENTICITY & ENGAGEMENT:
47          - You have a authentic personality, you are really
               kind and you also use emojis sometimes to be more
               authentic.
48          - Ask smart counter-questions when necessary to give
               precise answers.
49          - If the context retrieved does not provide a
               sufficient answer, state:
50            "I do not have enough relevant information to
                 answer your question."
51          - Always ensure accuracy by using the provided **
               context** below.
52
53          \n\n
54          Context: {context}
55          """
56      )
57
58      # Step 2: Create the ChatPromptTemplate with placeholders
            for the chat history and the user input
59      qa_prompt = ChatPromptTemplate.from_messages(
60          [
61              ("system", system_prompt),          # System
                   message with IT support assistant rules
62              MessagesPlaceholder("chat_history"),  #
                   Placeholder for chat history
63              ("human", "{input}"),                #
                   Placeholder for the users input/question
64          ]
65      )
66
67      # Step 3: Create the question-answer chain using the
           language model and the QA prompt
68      question_answer_chain = create_stuff_documents_chain(
           get_lmm(), qa_prompt)
69
70      # Step 4: Create the retrieval chain with a history-aware
            retriever that uses the contextualized prompts
71      rag_chain = create_retrieval_chain(
           history_aware_retriever_with_contextualization(),
           question_answer_chain)
72
73      # Step 5: Return the complete chain, which combines
           retrieval and question-answering
74      return rag_chain
```

Listing 3 RAG Chain for the IT Support Assistant

4.2.4 Frontend Logic

In the final step of the RAG pipeline, the system is integrated into a frontend interface, enabling users to interact with the virtual assistant through a chat-based application. The implementation utilizes the Streamlit framework, which provides convenient components for building interactive user interfaces, including support for rendering chat messages and input fields. To enable continuous dialogue with the assistant, the chat history is managed via the *st.session_state* object, which allows messages to persist across user interactions. Each new message, whether from the user or the assistant, is appended to the conversation history and displayed on the UI, thereby maintaining context throughout the session. A custom function named *generate_response()* is used to handle the response generation. This function calls the RAG pipeline asynchronously and streams the assistant's reply token by token, providing a more natural and real-time chat experience. Furthermore, each generated answer gets a reference link to the most relevant source document at the end of the response message, thereby enhancing transparency and trust by enabling traceability in the provided information.

Figure 5 illustrates the resulting user interface. The design is intentionally kept minimalistic to focus on usability and clarity. The welcome message appears at the top, while users can enter queries at the bottom. The application is accessible via: https://rag-thesis.uni-muenster.de/

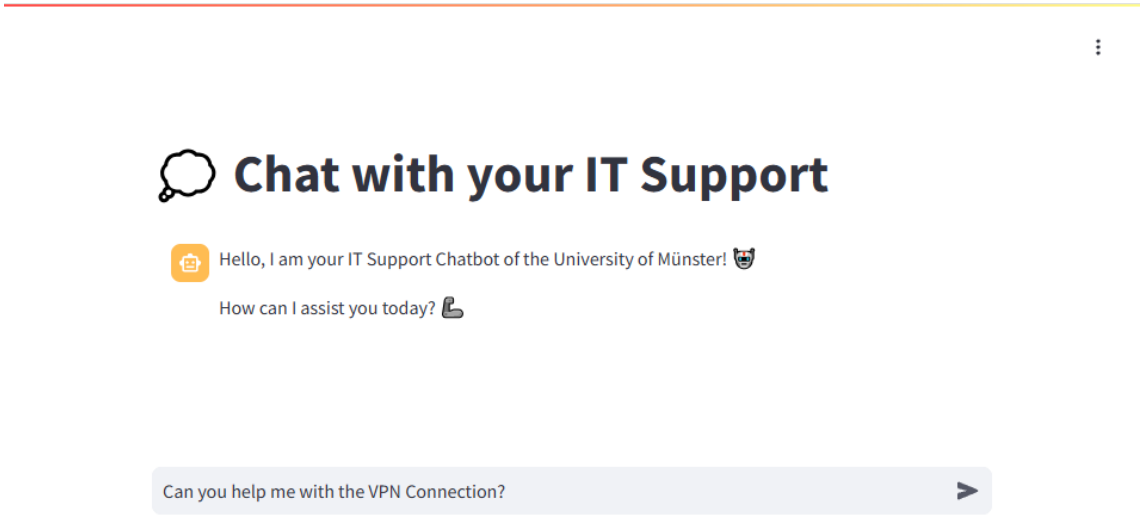


Figure 5 User interface of the RAG-based chatbot

Although Streamlit offers various options for customization and advanced UI design, a simplified layout was deliberately chosen for this implementation and subsequent

evaluation. Nevertheless, the application could be extended further by embedding it into the official website of the CIT using Streamlit's embedding functionality, thereby enabling its direct use in production (Snowflake Inc., 2025c). In this project, however, the application was containerized using Docker and deployed on the Kubernetes cluster maintained by the University of Münster, which is discussed in the following section.

4.3 Deployment on Kubernetes

The decision to deploy the app on the Kubernetes cluster of the University of Münster came from the motivation that with project completion the CIT has a simple takeover of the chatbot and has a productive configuration that they can orchestrate and scale responsibly. Kubernetes is an open-source platform designed to automate the deployment, scaling, and management of containerized applications (The Linux Foundation, 2025a). It offers a reliable and efficient way to orchestrate services by providing features such as service discovery, load balancing, self-healing, and automated rollouts and rollbacks (The Linux Foundation, 2025b). The advantages of Kubernetes lie in its ability to manage complex application lifecycles, ensure high availability, and optimize resource usage across multiple environments (The Linux Foundation, 2025c). It is especially useful in large-scale environments where microservices and container-based architectures are employed (The Linux Foundation, 2025c).

The University of Münster hosts its own Kubernetes infrastructure through the CIT, offering a robust and shared platform for running containerized services securely, observably, and efficiently (University of Münster, 2025a). This infrastructure supports workgroups and departments by offering this service for projects enabling scalable deployments. The University's Kubernetes cluster is composed of several central components (University of Münster, 2025b): Cilium is used as the Container Network Interface (CNI). It manages internal and external network connections between pods and services, implements firewall policies, and handles outbound traffic routing. Istio acts as both the ingress and egress gateway, controlling HTTP(S) traffic in and out of the cluster. As a service mesh, it provides advanced features like mutual TLS, load balancing, and support for multi-cluster setups. Gatekeeper enforces security policies through the Open Policy Agent (OPA). It validates resources upon creation, ensuring compliance with organizational policies, such as setting resource limits or applying default configurations. The University operates its Kubernetes environment across three different multi-cluster setups (University of Münster, 2025c): dev, staging, and prod. While tenants have access to the staging and production

clusters, the development cluster is reserved for testing the core cluster functionalities. Tenants can use the staging environment to test their own applications before going live in the production environment. These clusters are distributed across two physical locations in Münster: ms1 (Einsteinstraße), which serves as the primary location, and ms2 (Schlossplatz), which is mainly used for backups and multi-cluster functionality.

As part of this infrastructure, the project was granted a dedicated namespace within the productive cluster to deploy and run the RAG-based chatbot application. This enables scalable and secure hosting of the application in a real-world environment, making it accessible to end users while benefiting from the orchestration, monitoring, and policy enforcement features provided by the University's Kubernetes platform.

Before setting up the deployment.yaml file a docker image needs to be created out of the application. For that the following Dockerfile was created:

```
# Use a slim Python base image built on Debian/Ubuntu for a small footprint
FROM python:3.11-slim

# Set the working directory inside the container to /app
WORKDIR /app

# Copy the dependency file into the container and install required Python packages
COPY requirements.txt requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application files into the container
COPY . .

# Expose port 8501 to allow access to the Streamlit app from outside the container
EXPOSE 8501

# Add a health check to verify if the Streamlit app is running and healthy
HEALTHCHECK CMD curl --fail http://localhost:8501/_stcore/health

# Define the default command to run the Streamlit app on container startup
ENTRYPOINT ["streamlit", "run", "app.py", "--server.port=8501", "--server.address=0.0.0.0"]
```

Listing 4 Dockerfile for Containerizing the Streamlit Application

The Dockerfile defines the setup for containerizing a Streamlit-based application. It starts from a minimal Python 3.11 image to ensure a lightweight build. The working directory is set to */app*, where the application files will reside. It then installs all Python dependencies required by the application, such as Streamlit, LangChain, ChromaDB and Beatifulsoup, listed in requirements.txt, followed by copying the entire project into the container. The application is configured to run on port 8501, which is the default for Streamlit, and this port is explicitly exposed to allow external access. A *HEALTHCHECK* is included to ensure the service is responding properly. Finally, the container starts the Streamlit application via the *ENTRYPOINT*, binding it to all network interfaces to ensure accessibility within a Kubernetes cluster or from the host machine. An image was built from this Dockerfile and subsequently uploaded to Docker Hub (https://hub.docker.com/r/josefsdocker/rag$_c$*hatbot*, making it accessible for deployment within the Kubernetes cluster via the deployment.yaml configuration.

To deploy the RAG-based chatbot on the University of Münster's Kubernetes cluster, following five YAML files were created and configured: deployment.yaml, service.yaml, gateway.yaml, networkpolicy.yaml, and virtualservice.yaml. These files define the necessary components for orchestrating the application within the productive cluster namespace *rag-thesis* assigned to the project.

The deployment.yaml file is responsible for defining how the application should be deployed. It specifies the Docker container image to be used, named *josefsdocker/rag_chatbot:latest*, the number of pod replicas, in this case equals one, and includes resource requests and limits to ensure efficient resource usage. Furthermore, it defines liveness and readiness probes that monitor the health and availability of the Streamlit app. The deployment also includes metadata and labels for identification and organization within the Kubernetes cluster.

The service.yaml file exposes the deployed application internally within the cluster via a ClusterIP service, making it accessible to other Kubernetes resources. It forwards traffic on port 8501 to the corresponding application pod.

The gateway.yaml file defines an Istio Gateway resource that enables ingress traffic from outside the cluster. It handles both HTTP and HTTPS traffic and sets up TLS termination for secure communication, using a certificate managed by the university.

The networkpolicy.yaml file, implemented using Cilium, enforces strict ingress and egress rules. It allows traffic to the app only from the Istio ingress gateway on the specified port and enables DNS resolution and optional internet access.

The virtualservice.yaml is used to define how incoming requests from the Istio gateway should be routed to the internal service. It acts as a routing layer, ensuring that external HTTP(S) traffic is directed to the correct pod based on host and path rules.

Lastly, to secure the application via HTTPS, a TLS certificate with the Common Name (CN) rag-thesis.uni-muenster.de, issued by the GEANT Certification Authority, was configured. The certificate was set up and integrated into the Kubernetes cluster by a support staff member of the CIT. It uses TLS version 1.3 and is valid until April 2, 2026.

Together, these files ensure a secure, scalable, and well-orchestrated deployment of the chatbot application on the University of Münster's Kubernetes infrastructure. This deployment builds the foundation for productive testing of the RAG-based chatbot. Thus, the next chapter focus on the evaluation of the chatbot from the user, IT staff and performance perspective.

5 Evalution of the Chatbot Performance

This chapter presents the evaluation of the developed IT support chatbot. The assessment is based on three separate surveys, each addressing different aspects of the chatbot's performance and utility. The first survey comprises a user study designed to assess two main aspects: (1) how the chatbot performs in comparison to other information retrieval approaches such as web search, LLMs, and the official CIT website; and (2) how users perceive the interaction with the chatbot in terms of usability, usefulness, and satisfaction. Furthermore, it investigates whether the chatbot is suitable for broader deployment and identifies any remaining limitations or areas for improvement. The second survey targets IT support staff, aiming to determine whether the chatbot can deliver responses similar in quality to those provided by human personnel. Additionally, the survey explores the potential of the chatbot to be integrated into first-level support processes, its contribution to improving efficiency or reducing operational workload, and whether further enhancements are necessary before deployment. The third survey focuses on the chatbot's accuracy by comparing its responses to the eight most frequently asked IT support questions with the official information provided on the CIT website. This comparison aims to identify inconsistencies or deficiencies in the chatbot's ability to deliver reliable answers. In the following sections, each of the three surveys is described in detail, followed by an analysis and presentation of the respective results.

5.1 User Study

To evaluate the chatbot from a user perspective, a questionnaire was designed using Microsoft Forms. The full questionnaire is provided in Appendix A. A total of 25 participants, primarily students or former students, took part in the study. The survey takes approximately 20 minutes to finish. Users were allowed to answer in German or English.

The survey is divided into two main parts. In the first part, participants were instructed to utilize different search strategies (including a general web search, an arbitrary LLM, the CIT website, and the developed chatbot) to find answers to a frequently asked IT-related question in the context of the University of Münster. Participants were then asked to assess whether each approach provided a correct answer in their opinion. Following this, they were required to rank the search approaches in terms of convenience, speed, and quality. Additionally, participants described their typical search behaviour when seeking answers to IT-related problems and reflected on the perceived advantages and challenges of each method. The

primary goal of the first part is to evaluate the chatbot's performance in comparison to conventional search methods and to gain insight into how users generally approach problem-solving for IT-related issues, including the difficulties they encounter. Which can shed more light on user requirements in synthesis. In the second part of the survey, participants engaged more deeply with the chatbot by using it to search for answers to either predefined frequently asked questions or their own individual queries. The aim of this section was to explore the chatbot's strengths and limitations and to collect user feedback on potential improvements. In the following, the results of the survey are analysed and presented in detail.

The first question asked if the search approach was able to provide a correct answer to a frequently asked question question. The results show that the IT Support Chatbot was perceived as the most effective tool, with 24 out of 25 participants indicating that it provided a correct answer to their question. This was followed closely by the CIT website (22 responses), then web search (19 responses). The arbitrary LLM was the least effective, with only 12 participants stating it provided a correct answer. Overall, the chatbot outperformed all other approaches in terms of answer accuracy.

The next three questions asked to rank the approaches in convenience, speed and quality. For convenience the results indicate that the IT Support Chatbot was most frequently ranked as the most convenient method to find answers to IT-related questions, with 72% of respondents selecting it as their first choice. Web search was generally placed in second or third position, while LLMs received varied rankings across the middle positions. The CIT website was most commonly ranked as the least convenient approach. In terms of speed, the IT Support Chatbot was considered the fastest approach, with 56% of participants ranking it first and 32% ranking it second. Web Search followed in second place, often being selected as either the first or second fastest method. LLMs were generally positioned in the middle ranks, while the CIT website was most frequently rated as the slowest option. Regarding response quality and accuracy, the IT Support Chatbot was again rated highest, being most often ranked first. The CIT website received mixed responses but tended to be placed higher than LLMs and Web Search in terms of correctness and detail. LLMs and Web Search were typically rated as less reliable, with Web Search most frequently being ranked last.

The fifth survey question asked participants to elaborate on how they would search for an answer to an IT-related issue in the context of the University of Münster. Since this was an open-ended question, a content analysis was conducted to identify and classify common search strategies. The analysis involved two main steps. First, the

responses were read and interpreted to derive meaningful categories that represent different approaches to seeking IT support. In the second step, a colour coding scheme was applied to the responses, visually highlighting the relevant parts of the answers according to the assigned categories. The whole answers and categorization for each response can be found in the appendix 9 and 10.

The derived categories reflect the diversity of information-seeking behaviour among participants. In particular, the category Ask Colleagues / Communication refers to respondents who prefer informal help channels such as talking to peers or colleagues. The Hybrid Strategy category represents cases where multiple approaches are mentioned or used in sequence (e.g. searching the CIT website first and then using Google if no result is found).

The following colour-coded legend was used to annotate the responses:

C **IT Support Chatbot** – Direct use or preference for the chatbot introduced in the survey.

G **Google / Web Search** – General web search using search engines like Google.

W **CIT Website** – Direct navigation through the official CIT or university website.

L **LLM (e.g. ChatGPT, UniGPT, etc.)** – Use of general LLMs not limited to university scope.

H **Ask Colleagues / Communication** – Peer-to-peer communication.

M **Hybrid Strategy** – A combination of the above, typically sequential.

The colour coding revealed that the most frequently mentioned search strategies were the use of Google or general web search (G) and direct consultation of the CIT Website (W), each appearing in 16 out of 25 responses. These were followed by the IT Support Chatbot (C), which was referenced in 12 responses. Additionally, peer-to-peer communication channels such as asking colleagues or using internal communication tools, such as a mattermost channel (P) were mentioned in 3 responses, while general-purpose large language models like ChatGPT (L) appeared in 2 cases. Furthermore, 17 out of 25 participants mentioned multiple approaches in their answers, indicating a clear tendency toward hybrid search strategies that combine sources like Google, the chatbot, and the official university website in sequence or as alternatives.

The open-ended responses to Question 6 were analysed using qualitative content analysis to identify recurring themes regarding the perceived advantages and challenges of the four evaluated approaches: Web Search, LLM, CIT Website, and IT Support Chatbot. All answers and the categorization can be found in the appendix 11,12,13,14 and 15. Several core categories emerged from this analysis: **Speed and quick access to information** was one of the most frequently mentioned advantages, particularly in reference to the IT Support Chatbot (e.g. ID 1, 3, 5, 8, 9, 19), which was praised for delivering prompt responses and actionable advice. Web search was also described as a fast method to reach the correct subpage directly (ID 9, 15), although some users noted that this speed came at the cost of completeness or contextual accuracy (ID 1, 11). Another prominent category was the **accuracy and reliability of information**. The IT Support Chatbot was frequently perceived as both accurate and detailed (ID 3, 5, 12, 19), with some users highlighting its ability to provide precise instructions tailored to the university's infrastructure (ID 1, 18). However, general-purpose LLMs were often criticized for offering vague or factually incorrect content (ID 7, 12, 17, 18, 22). In contrast, the CIT website was valued as a trusted source with up-to-date information (ID 10, 15), but a website that required significant effort to navigate. **Ease of use and convenience** formed another critical category. The Chatbot was generally regarded as intuitive and user-friendly (ID 1, 8, 14, 21), while LLMs were recommended for their simplicity in handling general queries (ID 1, 11, 15). Conversely, navigational challenges were reported in connection with the CIT Website (ID 2, 6, 8, 19, 21, 24) and, to a lesser extent, the web search approach (ID 5, 7, 20). These challenges included difficulties locating the appropriate subpages, unclear menu structures, or uncertainty regarding the correct search terms. In terms of **level of detail and presentation**, the Chatbot stood out positively for offering structured, step-by-step instructions (ID 6, 17, 23), while the CIT website received praise for its inclusion of screenshots and broader service overviews (ID 16, 19). However, some participants also desired better integration of visuals or the possibility to submit screenshots to the Chatbot (ID 16). **University specific information** was a distinct advantage associated with the Chatbot and CIT Website. Multiple responses emphasized the value of accessing information aligned with the university's domain and wording by University of Münster (ID 5, 11, 18). LLMs and web search, in contrast, were sometimes perceived as too general or not tailored to the university context (ID 13, 22, 23). Other emerging categories included **trust and perceived reliability**, with several users expressing hesitation to fully trust LLMs (ID 7, 17, 19), and in a few cases, also the Chatbot (ID 9, 22). Source verification and link availability were noted as limitations, especially for LLMs, which often failed to provide URLs or references for further validation (ID 1, 7, 12). Finally, design-related aspects were addressed;

users mentioned aesthetic or usability benefits of the Chatbot and CIT Website (ID 1, 19), but also noted isolated issues such as security warnings or access barriers when using the Chatbot (ID 1). Overall, the analysis demonstrates that the IT Support Chatbot was widely perceived as a fast, accurate, and university-specific tool. However, challenges remain regarding its visibility and trustworthiness, as well as feature suggestions for further interaction. Comparatively, while the CIT Website offers official and reliable content, its navigation and discoverability continue to pose usability challenges. LLMs and web search approaches were valued for their speed and accessibility but fell short in providing university-specific, reliable, and well-structured responses.

The second part of the user study focused on the targeted evaluation of the chatbot itself. Participants were first presented with a series of statements assessed on a 5-point Likert scale, followed by open-ended questions to gather more nuanced insights into the chatbot's perceived viability as an IT support solution. The Likert-scale items were adapted from the evaluation framework developed by Essop et al. (2023), who designed a questionnaire specifically for university FAQ chatbots. Their approach was grounded in a comprehensive literature review and structured using the extended Unified Theory of Acceptance and Use of Technology (UTAUT2) framework (Tamilmani et al., 2021). Therefore, the evaluation encompassed a range of chatbot design dimensions, including Usability, User Interface (UI), Natural Language Processing (NLP), Anthropomorphism, Personas, User Experience (UX), Efficiency, Quality and Accuracy, and Conversational Memory. In this context, Anthropomorphism refers to the degree to which a chatbot exhibits human-like characteristics such as emotional expression, social presence, and likability, which can influence user engagement and acceptance. (Essop et al., 2023). Personas, on the other hand, describe the stylistic and personality elements embedded into the chatbot, such as the use of emojis, humour, and tone, which collectively shape the overall interaction experience (Essop et al., 2023). Building on this evaluation model, the questionnaire was adapted to the specific context of IT support, and additional focus was placed on metrics particularly relevant for a RAG-based chatbot, namely Efficiency, Quality and Accuracy, and Conversational Memory. The questions with their assigned metric dimension can be looked at in the appendix 16. Participants rated each item using a 5-point Likert scale ranging from *Strongly Agree* to *Strongly Disagree*. The results are presented in table 1, which displays the absolute number of responses and percentage for each level of agreement across all evaluated questions.

The Likert-scale evaluation revealed that a majority of respondents perceived the chatbot's responses as complete and accurate, with 84% selecting *agree* or *strongly*

agree. Most users also found the chatbot to be quick in delivering answers, with 88% expressing agreement. Regarding the chatbot's ability to retain conversational context, 79% of participants agreed or strongly agreed. The perceived human-likeness of the chatbot's responses showed a broader distribution, with 60% agreeing or strongly agreeing. Similarly, 68% indicated that the chatbot's conversational style positively influenced their willingness to use it in the future. When asked whether the chatbot reduced the effort needed compared to other search methods, 72% responded *agree* or *strongly agree*. A smaller majority, 64%, stated they would prefer using the chatbot over other sources in the future. Ease of use received strong affirmation, with 96% selecting *agree* or *strongly agree*. The visual appearance of the chatbot was perceived positively by 64% of respondents, while 76% agreed that the chatbot's design and personality made the interaction more engaging. Lastly, 92% expressed satisfaction with the overall experience of using the chatbot.

The next two question asked the user what they liked or disliked about the chatbot. A content analysis revealed that the most appreciated features were the chatbot's clear and structured responses, the inclusion of reliable links to official university sources, and the general speed and convenience of interaction. Many users highlighted the chatbot's ability to ask clarifying questions and tailor its responses accordingly, as well as its friendly tone and ease of use which is especially valuable for less experienced users. On the other hand, the most frequently reported drawbacks included the slow token generation animation, occasional inaccuracies or incomplete answers, and a perception of excessive talkativeness that did not align with user expectations for an IT support tool. Some participants also voiced concerns about trust in LLMs, limited memory across sessions, or a lack of customization options in how the chatbot interacts. The detailed answers are in the appendix 17,18, 19 and 20 presenting the feedback category as well.

The evaluation continued with question 10 and 11 to explore the preferences of participants in using the chatbot compared to self-directed search or human support. The answers are summarized in table 2 and 3. In response to Question 10, a significant number of users emphasized the chatbot's convenience, speed, and structured responses as key reasons for preferring it over manual search, especially when seeking step-by-step IT instructions. Many participants noted they would use the chatbot for specific or unfamiliar issues, while still resorting to traditional search for quick or habitual access. A subset of users expressed hesitancy due to trust concerns or the chatbot's current access setup, indicating they would use it more often if it were more integrated into the university's web environment. Regarding Question 11, users generally indicated a strong preference for using the chatbot for

common and straightforward IT problems. The main benefits cited were availability, the non-judgmental nature of interaction, and reduced response time compared to human support. Nonetheless, when dealing with complex, personal, or approval-based issues, users still valued human expertise and preferred direct contact. Several participants also indicated that they would try the chatbot first and only escalate to human support when necessary, suggesting its potential as a valuable first-level support tool. The detailed answers of the users are available in appendix 21,22, 23 and 24.

The final question of the user study invited participants to provide suggestions for improvement or report any issues encountered with the chatbot. Table 4 present the major limitations encountered and suggestions for chatbot by the users. The majority of respondents provided either no suggestions or expressed satisfaction with the chatbot's performance, citing no major problems. However, several participants proposed enhancements to improve usability and accuracy. One frequently mentioned issue was the lack of conversation history, which required users to repeat their queries upon reopening the chatbot. Additionally, respondents expressed interest in expanding language support, such as including Spanish for international students even though the chatbot is capable of speaking in Spanish. Others recommended user interface improvements, including the option to upload screenshots and select clickable response categories instead of typing, to streamline interaction. A subset of participants noted inaccuracies or confusion in the chatbot's answers, particularly concerning technical distinctions between university services like the IT portal and the intranet. These users emphasized the importance of reliable sources and alignment with official university content to maintain trust. Finally, suggestions were made to provide faster access to fallback responses like "I do not know" and to offer more technical, minimal interfaces for experienced users. Overall, the feedback highlights a desire for both enhanced interactivity and greater content reliability to support a broader range of user needs. All 18 responses can be found in detail in appendix 25 and 26.

Statement	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
The chatbot's responses were complete and accurate to answer my questions.	9 (36%)	12 (48%)	3 (12%)	1 (4%)	0 (0%)
Using the chatbot to find my answer felt quick.	13 (52%)	9 (36%)	2 (8%)	1 (4%)	0 (0%)
I felt that the chatbot was able to retain context and remember previous messages throughout the conversation.	6 (25%)	13 (54%)	3 (12%)	2 (8%)	0 (0%)
I believed the chatbot understood and responded to me like a human would when answering questions.	5 (20%)	10 (40%)	5 (20%)	4 (16%)	1 (4%)
The chatbot's conversational style influenced my willingness to use it in the future.	8 (32%)	9 (36%)	6 (24%)	2 (8%)	0 (0%)
The chatbot allowed me to resolve my IT issue with less effort compared to web search, LLMs, or the CIT website.	11 (44%)	7 (28%)	4 (16%)	3 (12%)	0 (0%)
From now on, I would prefer the chatbot over other ways to obtain information on my IT-related problems.	9 (36%)	7 (28%)	5 (20%)	3 (12%)	1 (4%)
It was easy for me to use the chatbot to get my question answered.	16 (64%)	8 (32%)	0 (0%)	1 (4%)	0 (0%)
The visual appearance of the chatbot made it easier for me to interact with it.	6 (24%)	10 (40%)	9 (36%)	0 (0%)	0 (0%)
The chatbot's design and personality made it more engaging to interact with.	10 (40%)	9 (36%)	6 (24%)	0 (0%)	0 (0%)
I was satisfied with the overall experience of using the chatbot.	15 (60%)	8 (32%)	1 (4%)	0 (0%)	0 (0%)

Table 1 Evaluation of Question 7 on the Likert-Scala

Category	Description	Respondents
Faster and Easier to Use	Chatbot is perceived as quicker and less effortful than manual search (Google/CIT)	1, 2, 3, 8, 15, 18, 19, 23
Helpful for Complex or Unfamiliar Issues	Particularly useful when users are unsure about their issue or need detailed, guided help	4, 5, 11, 13, 14
Conditional on Reliability or Context	Use depends on whether the chatbot is considered reliable or equally official as the CIT website	6, 7, 16, 17, 21
Preference for Manual Search	Preference for web search or CIT website due to trust, habit, or usability	9, 10, 12, 20, 22

Table 2 Categorization of Responses to Question 10: Use of Chatbot Instead of Searching on Your Own

Category	Description	Respondents
Preferred for Simple/Standard Issues	Chatbot suitable for FAQs, step-by-step guides, or known solutions	1, 3, 4, 5, 6, 7, 9, 10, 11, 13, 14, 15, 17, 18, 21, 23
Human Support Preferred for Complex or Sensitive Issues	Human interaction is seen as necessary when dealing with complex, sensitive, or approval-related matters	2, 4, 6, 10, 11, 16, 19, 20, 22
Chatbot Accessibility and Emotional Comfort	Chatbot valued for being non-judgmental, always available, and easier to approach than people	1, 8, 12, 14, 18, 21, 23
Conditional Use	Chatbot used as a first point of contact before escalating to human support if needed	12, 13, 15, 17, 19, 23

Table 3 Categorization of Responses to Question 11: Use of Chatbot Instead of Human Support

Category	Description	Respondents
Persistence and History	A user wishes for conversation history or session persistence to revisit past responses.	1
Multilingual Support	Suggestion to support more languages, e.g. Spanish, for international students.	4
Token Generation Animation	A user finds the typing animation slow or distracting.	5
Incorrect or Misleading Information	Reported instances of factual inaccuracies or unhelpful sources.	7, 8, 17
UI or Interaction Improvements	Users suggested clickable response options and better input efficiency.	9, 14
Media Upload and Visual Enhancements	Suggestions to allow uploading screenshots and include more images from official instructions.	12, 13
Reliability of Sources and Answer Completeness	Desire for more consistent source attribution and more complete responses.	12, 15, 17
Limited on CIT content	A user reported misleading response becuase of the wrong context.	17
No Suggestions or Positive Feedback	Respondents who explicitly stated no suggestions or were satisfied.	2, 3, 6, 10, 11, 16, 18

Table 4 Results of the Content Analysis of Suggestions and Issues (Question 12)

5.2 IT Support Staff Study

To evaluate the chatbot from a IT Support staff perspective, a questionnaire was designed using Microsoft Forms. The full questionnaire is provided in appendix B. A total of 10 participants from the IT Support staff department participated. The survey takes approximately 15 minutes to finish. The staff were allowed to answer in German or English. The task given by the questionnaire it to test the RAG-based chatbot's ability to function as a 1st-Level IT support assistant by asking IT-related questions. Afterwards they should complete the questionnaire by answering the questions.

The first questions use a 5-point Likert scale similar to the user study. Again the questions are based on metric dimensions but this time focusing on the chatbot's capability on improving the 1st-Level Support operations. The metrics include, accuracy, chat history, anthropomorphism, personas, operational efficiency, usability, user interface, availability, adoption, operational efficiency and user experience. In appendix 27 the assignment of metrics to questions are illustrated.

The table 5 displays the absolute number of responses and percentage for each level of agreement across all evaluated questions. Regarding accuracy, 70% of respondents either agreed or strongly agreed that the chatbot delivers complete and accurate responses, while 20% disagreed. Context awareness was also positively received, with 80% affirming that the chatbot could retain context during conversations. In terms of anthropomorphism, 60% agreed or strongly agreed that the chatbot responded like a human, though 30% remained neutral. When asked if the chatbot's conversational style resembled a real IT support agent, 40% agreed, 40% remained neutral, and 20% disagreed. The chatbot's ability to resolve IT issues like a human support agent was supported by 50%, while 30% expressed disagreement. In terms of integration into 1st-level support, 50% agreed it could play a role, though 30% remained neutral. Half of the respondents also agreed that the chatbot could help relieve personnel shortages, even though 30% disagreeing. Regarding user adoption, opinions were mixed, as only 40% agreed the chatbot would achieve widespread adoption, while another 40% remained neutral. The chatbot's personality and design were positively rated by 60%, but one person found it disengaging. In terms of availability, 70% agreed the chatbot could increase support accessibility. Similarly, 70% believed the chatbot simplifies the process of finding information on the CIT website. Furthermore, 80% of IT staff agreed the chatbot could reduce repetitive requests. However, when asked whether the chatbot could fully replace the support

hotline, 80% either disagreed or strongly disagreed, clearly indicating that while the chatbot adds value, it is not yet seen as a complete substitute for human IT support.

The next two question this time asked the IT staff what they liked or disliked about the chatbot. The feedback from the IT staff highlighted several positive aspects: Respondents (R) especially appreciated the speed and availability of the system, which enabled fast access to frequently requested information and support at any time of the day (R3, R4, R5). In addition, the chatbot's ability to deliver structured, step-by-step recommendations was commended, particularly for complex or less common IT problems (R1, R5, R6). Someone also valued the inclusion of contact information for the relevant follow-up support of the CIT (R7) and found the design and interface to be appealing (R8). Someone also appreciated the endurance of the chatbot keeping always the same mood independent of the frequencies of the questions (R2).

However, several limitations were also noted. Some IT staff criticized the interaction flow, finding it unintuitive when responding to multi-part queries (R1), and mentioned the chatbot's dependence on specific keywords to generate meaningful answers (R2). Others pointed out the issue of overgeneralization (R5) and were particularly concerned about the chatbot's tendency to deliver confident but incorrect technical responses, which could potentially mislead users (R6). Additionally, complaints were raised about the chatbot's use of irrelevant sources (R7) and its susceptibility to hallucinations, often requiring multiple interactions before receiving a useful response (R8). The detailed answers can be found in appendix 28 and 29.

The next questions deals with the ability of the chatbot to resolve support requests. The responses reveal a consensus among IT staff that the chatbot is well-suited to resolve standard, clearly defined, and frequently recurring support requests, such as password resets or general navigation questions (R1-R6). However, several participants emphasized the chatbot's current limitations in handling complex, context-sensitive, or user-specific cases, particularly those requiring backend system access or up-to-date operational data (R1, R5-R7). Requests such as advanced identity verification or system incidents (e.g. software availability) were cited as remaining within the domain of human support. Additionally, one respondent highlighted that users often struggle to articulate their problems accurately, presenting another limitation for chatbot-based interactions (R7). The detailed answers can be found in appendix 30.

The second last question deals with the economical or operative effects on the IT Support. The IT staff identified a range of potential economic and operational improvements that could result from integrating the chatbot into 1st-level IT support. A key advantage noted by several respondents (R1, R3, R6, R7) is the chatbot's ability to provide 24/7 support, increasing service availability beyond typical office hours. Others emphasized that delegating standard questions to the chatbot could reduce the burden on human support personnel, allowing them to focus on more complex issues (R4, R5, R7). This redirection of simple tasks could also lead to quicker responses and fewer repetitive questions (R2, R4, R5, R7). Additionally, participants pointed out that some users, particularly students, may prefer interacting with a chatbot over contacting human support due to comfort and convenience or changing communication preferences (R2, R7). However, some concerns were raised: one respondent (R6) questioned whether the chatbot would yield actual resource savings, as time gained might be offset by correcting chatbot errors. Another one (R1) was unsure if the chatbot offers real advantages compared to existing FAQs. Finally, one critical respondent (R8) expressed concerns about the possible negative social consequences of automation, such as job displacement and decreased customer satisfaction. Detailed answers are presented in appendix 31.

The final question asked to share suggestions for improvement and to identify missing features or significant limitations. A recurring recommendation concerned the extension of the chatbot's data coverage (R1, R3, R5, R6), particularly the inclusion of training data for services such as Sciebo and domain-specific resources like the HPC cluster documentation. Participants emphasized that broader access to relevant sources would significantly enhance the chatbot's utility in support contexts. Another point raised was the lack of conversational memory or context awareness (R2). The respondent suggested that the chatbot could improve by recognizing when users develop partial solutions themselves during dialogue and incorporating this feedback into future responses. One respondent noted an issue with platform-specific output (R4), describing a situation in which the chatbot failed to differentiate between operating systems, though this problem could not be replicated. Finally, concerns were expressed regarding the transparency and accuracy of cited sources (R5), with the chatbot sometimes attributing answers to seemingly arbitrary or unrelated pages. This led to a recommendation for clearer indication of data origins and limitations, especially when the chatbot lacks access to certain documentation. The full answers are in appendix 32.

Statement	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
The chatbot is able to provide complete and accurate responses to IT-related questions.	1 (10%)	6 (60%)	1 (10%)	2 (20%)	0 (0%)
The chatbot was able to retain context and remember previous messages throughout the conversation.	2 (20%)	6 (60%)	1 (10%)	1 (10%)	0 (0%)
The chatbot understood and responded like a human would when answering questions.	3 (30%)	3 (30%)	3 (30%)	1 (10%)	0 (0%)
The chatbot's conversational style is similar to a real IT support person.	0 (0%)	4 (40%)	4 (40%)	2 (20%)	0 (0%)
The chatbot is capable to resolve IT issues like the human support personnel.	0 (0%)	5 (50%)	2 (20%)	2 (20%)	1 (10%)
I believe that the chatbot can be part of the 1st-Level Support from now on.	1 (10%)	4 (40%)	3 (30%)	2 (20%)	0 (0%)
I believe that it could relieve or compensate for personnel shortages.	0 (0%)	5 (50%)	2 (20%)	2 (20%)	1 (10%)
I believe the chatbot will have a huge adoption of users.	0 (0%)	4 (40%)	4 (40%)	2 (20%)	0 (0%)
The chatbot's design and personality made it more engaging to interact with.	1 (10%)	5 (50%)	3 (30%)	0 (0%)	1 (10%)
The chatbot could be a beneficial way to increase the availability of IT Support.	2 (20%)	5 (50%)	2 (20%)	1 (10%)	0 (0%)
The chatbot provides an easier way to find sources on the CIT website than searching by yourself.	2 (20%)	5 (50%)	2 (20%)	1 (10%)	0 (0%)
Do you think the chatbot can help reduce the number of repeated requests of frequently asked questions?	2 (20%)	6 (60%)	1 (10%)	1 (10%)	0 (0%)
Do you think the chatbot is able to completely substitute the Support Hotline?	0 (0%)	1 (10%)	1 (10%)	2 (20%)	6 (60%)

Table 5 Evaluation of IT Staff Feedback on the Likert Scale

5.3 Performance Study

In this study the performance in factual accuracy and quality of the chatbot generated answers was evaluated. For that, eight frequently asked questions and their corresponding sources on the CIT webpage were given by the Head of IT Services for evaluation. The evaluation focused on whether the chatbot's responses accurately reflected the content of the official sources, whether the answers were complete, clearly formulated without ambiguity, and whether the correct source link was provided. For each criterion met, one point was awarded per response. The total score was then averaged across all questions to assess the chatbot's overall factual performance. Table 6 presents the results. The results show that the chatbot performed well overall, delivering fully correct, complete, and clear responses with accurate source attribution in five out of eight cases. In the case of the WLAN configuration question, the chatbot directly responded with instructions for the university's "unims" network, neglecting to counter question which of the available WLAN networks the user intended to configure. This lack of clarification affected the clarity score. Regarding the VPN setup, the response included information on how to start and stop a VPN session but omitted the installation guide. While the chatbot did ask whether the VPN client had already been installed and offered further assistance if not, this conditional approach led to a deduction in completeness. The question where the chatbot failed most significantly was related to the Remote Desktop connection. Although the chatbot correctly linked to the general information page, it failed to include or reference the installation instructions specific for different operating systems. This issue was traced back to a limitation in the link crawling, as the crawler did not access subpages containing the relevant instructions, based from the site map.

Question	**Factual Correctness**	**Completeness**	**Clarity**	**Correct Source**
How do I set up an OTP generator?	1	1	1	1
I forgot my password. What should I do now?	1	1	1	1
How do I set up my WLAN?	1	1	0	1
How do I establish a VPN connection?	1	0	1	1
How do I get access to Office 365?	1	1	1	1
I can't find the poster printer. What could be the reason?	1	1	1	1
How do I connect to the Remote Desktop?	0	0	0	1
How can I extend my university ID?	1	1	1	1

Table 6 Evaluation of Chatbot Responses for Frequently Asked Questions

6 Discussion

This chapter presents a critical discussion of the developed RAG-based chatbot in light of the initial design problem and the requirements identified from key stakeholders. The aim is to evaluate whether the implemented artifact successfully addresses the goals of improving IT support at the University of Münster by meeting stakeholder expectations regarding decreased workload for the IT staff and increased availability and correct answers by the IT Support for the students. The analysis begins with an examination of the results from the three evaluation perspectives: the user study, the IT staff feedback, and the performance evaluation of the chatbot in handling frequently asked questions. These findings are then systematically discussed in relation to the implemented system to assess its practical effectiveness. Based on this evidence, the chapter evaluates whether the originally defined design objectives have been met. Building on this analysis, implications for users, IT personnel, and the university administration are derived and contrasted with the expected effects formulated during the design phase. Based on these insights, actionable recommendations are proposed to guide further development. The chapter concludes by discussing the limitations of this study, including contextual constraints and methodological boundaries.

The goal of the first part of the user study was to find out if the RAG-based chatbot is able to provide factual correct answers faster and more conveniently compared to other search strategies or tools. The IT Support Chatbot was perceived by most participants as the most effective method for obtaining correct answers, indicating that the majority of student inquiries could be adequately addressed through the chatbot. In theory the CIT website should be able to provide an answer to all given questions to the users in the questionnaire. Still users often failed to locate the relevant content. There might be two causes why a user did not find the right answer on the CIT website, either the navigation to the question was not feasible or the user did not understand the full context of the question, because for example the question "How can I extend my university ID?" can have two answers depending on the context, either it refers to the general university ID which cannot be extended after exmatriculation or it refers to the extension of group memberships where someone can make an application in the IT-Portal. This highlights the necessity of contextual understanding in information retrieval. Because of the relevance in question understanding the chatbot was instructed in the system prompt to always to try to clarify the question by asking smart counter questions. For convenience, speed and quality the ranking of the respondents were as expected. On the one hand, the CIT website was ranked on top with the chatbot to provide qualitative answers, but when it comes to conve-

nience and speed the CIT website was voted on the last rank compared to the other search approaches. The IT Support Chatbot ranking one on all three dimensions suggesting to overcome the challenge of navigating through documents by directly retrieving the relevant information and provide it in a precise response. Again emphasizing the advantage of the RAG-based architecture providing the automation of retrieval of relevant documents. These findings were further supported by responses, where users frequently cited navigational difficulties with the CIT website. Another interesting concern of the users was the trust and perceived reliability. On the one hand, users were expressing hesitation to trust the generated answers by the LLMs and the chatbot. On the other hand, the ability of source verification by the chatbot reduced the scepticism. An important observation from the user feedback is the stated use of combined search strategies. Many respondents reported that they typically use a combination of approaches, such as performing a web search and then navigating to the CIT website. A key insight derived from the responses is that web search remains the most common method for finding information, primarily because it is perceived as fast and convenient. However, several participants indicated that they would prefer to use the chatbot, provided they were aware of its existence, it was already publicly available, and it was seamlessly integrated into the CIT website. This suggests that accessibility, visibility, and integration are critical factors for broader adoption of the chatbot. Another important insight from the user study concerns the level of detail and presentation of the chatbot's responses. Participants positively highlighted the structured, step-by-step nature of the answers by the chatbot. However, in contrast to the CIT website, some users criticized the lack of visual elements such as screenshots or the inability to upload error messages or images for context. This limitation originates from the current design of the developed RAG-based system, which is restricted to natural language processing (NLP) and does not support multimodal inputs. In the context of IT support, where many issues are visual or interface-related, this is a significant consideration. Nevertheless, modern RAG frameworks such as LangChain already support integration with multimodal LLMs, enabling the processing of text, images, and other media formats. While this flexibility theoretically allows for future extension of the system, it raises questions about cost-efficiency and practical implementation. On the one hand, incorporating multimodal capabilities would increase both the complexity of the pipeline and the operational costs, as image-based queries are significantly more expensive to process than plain text. On the other hand, including screenshots in tutorials would require additional preprocessing, embedding, and storage, adding further development effort. Given that the current system already delivers accurate and context aware responses with appropriate source links, which leading users to the original web pages that often contain visual guidance, there remains a trade-off

between enhanced user experience and the associated cost and implementation effort. Thus, while the system architecture supports multimodal extension, a careful cost-benefit analysis is needed before proceeding with such enhancements. Another noteworthy observation emerged from the user feedback, where one participant expressed the desire for the chatbot to support Spanish, particularly to assist Erasmus students. Interestingly, the chatbot is technically capable of responding in Spanish, as the underlying LLM includes robust translation capabilities. However, this functionality was not apparent to the user. By design, the chatbot automatically replies in the language in which the question is asked, as specified in the system prompt. Nonetheless, due to the absence of an introductory message or interface cues that clearly communicate this feature, users may assume that the chatbot is limited to English. This highlights an important usability issue. The capabilities of the chatbot must be explicitly communicated to users at the beginning of the interaction. A clearly formulated welcome message or short introductory statement could inform users about supported languages and clarify the chatbot's ability to answer domain-specific questions based on information from the CIT website. This would help manage expectations and empower users to make full use of the system's functionalities, thereby improving the overall user experience and reducing potential frustration caused by perceived limitations. Moreover, most users confirmed that the chatbot was able to retain chat history and context within a single conversation, which positively contributed to the perceived coherence and usability of the system. However, one limitation identified by a participant was the inability to access previous conversations after refreshing or revisiting the page. This meant that users had to repeat their queries if they returned to the chatbot later, potentially reducing efficiency and user satisfaction. Persisting chat history beyond the current session could enhance usability by allowing users to quickly reference previous interactions and avoid redundant queries. As the current implementation stores conversation data only within the active session, a possible enhancement would be to integrate a login mechanism for university members. This would enable the saving and retrieval of past conversations across sessions and devices, supporting more personalized and efficient interactions with the chatbot. However, such an expansion of the application would need to be carefully weighed against the anticipated benefits, as the actual frequency with which users revisit past conversations may be low and may not justify the additional development and resource investment. Another important point concerns the chatbot's potential for future adoption. This is supported by the fact that 68% of respondents indicated a willingness to use the chatbot again in the future, while 72% agreed that the chatbot required less effort to find the desired information compared to alternative approaches. Further, the chatbot's 24/7 availability was particularly appreciated, especially by users who work during evenings

or weekends and may not have access to human support increasing the motivation to use this chatbot. Additionally, many users highlighted that the chatbot allows them to ask follow-up questions without hesitation or fear of judgment, which lowers the entry barrier for IT-inexperienced individuals. The ability to interact naturally and receive immediate, context aware responses further supports its role as an effective first point of contact. At the same time, respondents acknowledged the continued importance of human support for complex, sensitive, or approval-dependent requests, positioning the chatbot as a valuable tool for initial assistance that helps reduce the workload of IT personnel by filtering routine questions. A critical observation from the evaluation is that the chatbot was unable to respond accurately to queries related to services beyond the scope of the CIT website, such as the university's research portal or other non-IT university services. In these cases, the chatbot occasionally produced hallucinated content or provided fallback responses like "I do not know" too late in the conversation. From this, it can be deduced that the current limitation of the knowledge base significantly restricts the chatbot's utility across broader university support contexts and that users like to ask questions beyond the CIT context. To address this, the scope of scraped documents must be expanded to include other relevant university platforms. Additionally, the prompt engineering of the RAG system should be refined to enforce earlier fallback responses when a question cannot be answered based on the available data. This would not only reduce hallucinations but also improve user trust. Furthermore, these issues may also originate from ambiguous phrasing or abbreviations in user queries, where the model without additional context might understand different meanings. In conclusion, extending the document base and improving the prompt design are necessary steps to enhance the accuracy and reliability of the chatbot in a diverse university environment. Therefore, ensuring the reliability of the chatbot's responses is of critical importance, particularly because there is a risk that users may interpret inaccurate answers as official advice. To mitigate this risk, it is essential to include a clear disclaimer at the beginning of the interaction, outlining the chatbot's role as a support tool rather than a definitive authority. Establishing transparent boundaries for the chatbot's capabilities helps manage user expectations and responsible use. Furthermore, users should be informed that the system currently relies on an external LLM, specifically gpt-4o-mini, which processes user inputs externally. This raises potential concerns regarding data sensitivity and privacy, especially when users unknowingly share personal or institutional information. A viable solution to this issue would be to integrate one of the locally hosted LLMs operated within the university's infrastructure. This would ensure greater control over data processing, strengthen compliance with internal data protection policies, and increase user trust in the system.

The primary objective of the IT support staff study was to assess whether the chatbot could effectively assist in 1st-level IT support by reducing the volume of repetitive tasks and contributing to improved operational efficiency. Overall, the chatbot's accuracy was positively rated, with the majority of respondents considering the responses complete and accurate. However, 20% of participants expressed disagreement, which can likely be attributed to instances where the chatbot was tested with queries beyond the scope of the CIT website. This again underscores a critical limitation: the current knowledge base is too narrow to accommodate the full range of support requests typically received by IT personnel. Staff members emphasized the need for integrating information from additional platforms such as Sciebo, which is frequently referenced in user inquiries. A key takeaway from the study is the consensus among IT staff that, while the chatbot shows strong potential for addressing routine and frequently asked questions, it is not suitable as a replacement for human support channels such as the IT Hotline. Rather, the chatbot is seen as a complementary tool which suits for resolving standard questions, but less capable of handling complex, system-specific, or backend-related issues that require human expertise. This distinction is crucial for understanding the chatbot's role within the broader IT support infrastructure. Nonetheless, one respondent expressed concern that the introduction of the chatbot could potentially lead to job displacement, suggesting that the university might prioritize cost-saving measures over maintaining high levels of customer satisfaction. However, this fear appears to be premature and not fully justified given the current capabilities of the system. As the chatbot is presently limited to answering standard inquiries based on information from the CIT website, its functional scope does not extend to handling complex, context-specific, or system-dependent issues that require human expertise. Moreover, the evaluation results clearly demonstrate that users continue to value and prefer human support for such cases. The strategic intent behind the chatbot is not to replace IT personnel but to relief their workload by automating responses to frequently asked questions. In this way, the chatbot serves as a supportive tool to optimize resource allocation and improve service availability, rather than as a substitute for human expertise. This insight underscores that RAG for LLMss in the context of IT support are not designed to fully replace human personnel as tasks such identification and admin system-access cannot be taken over by such a chatbot. Instead, their strength lies in domain-specific information retrieval, where they can efficiently provide accurate and context-relevant answers. From an economic perspective for the university the chatbot has the potential to reduce the workload of IT staff by handling repetitive, frequently asked questions, thereby enabling the redirection of human resources toward more complex tasks. However, to accurately assess the extent of these benefits, the chatbot must be deployed on a broader scale

and integrated into the daily operations of the IT support team. Only through real-world usage over time can its true impact on operational efficiency and cost savings be systematically evaluated, showing limitation of this study.

The results of the performance evaluation demonstrate that the chatbot was generally capable of retrieving and generating suitable answers for nearly all frequently asked questions. However, certain limitations emerged in cases where the user's query lacked sufficient context. For instance, the question "How do I set up my WLAN?" is inherently ambiguous, as the university offers multiple WLAN options. The chatbot, in this case, assumed that the user referred to the most commonly used network, "uni-ms", and provided instructions accordingly. While technically correct, this response lacked clarity, as it did not prompt the user to specify which WLAN they intended to configure, even though the retrieved documents contained information on alternative networks. Similarly, in response to the question "How do I establish a VPN connection?", the chatbot described the process of initiating and terminating a VPN session but omitted the installation procedure for the VPN client. Although it did indicate that installation is a prerequisite and offered to assist if the VPN was not yet installed, the omission affected the completeness of the response. These examples underscore the critical role of question formulation in the retrieval and generation process. In cases where the input query lacks specificity, the chatbot tends to rely on the most prominent or frequently accessed content from the retrieved documents, which may not fully address the user's intent. This highlights the importance of implementing mechanisms—either through prompt design or conversational clarification strategies—that enable the chatbot to ask for additional context when needed to enhance answer precision and relevance. To address these limitations, several techniques can be employed to enhance the performance of the RAG-based chatbot. One approach could be query rewriting, which reformulates user questions from different angles to better capture the underlying intent. In combination with improved prompt engineering, the system can be guided to proactively ask counter questions when a query is vague or context-dependent. This would enable the chatbot to collect additional clarifying information before attempting to generate an answer, thereby increasing both completeness and accuracy. These strategies highlight the potential for further RAG enhancements and represent valuable directions for future research and development. Lastly, one question could be solved at all because the relevant document could not be found to give the relevant context. This incident revealed that not all documents from the CIT website were successfully loaded into the vector store. It appears that either the sitemap did not allow crawling to this particular subpage, or that a logical error in the implementation excluded certain links from being processed. This issue must be addressed as a

priority. Therefore, the link scraping function should be thoroughly reviewed, and the set of retrieved links should be supplemented with the complete list of available pages. A simple solution might already lie in adjusting the base URL for scraping to https://www.uni-muenster.de/, which could ensure a more comprehensive coverage of the entire website and even more documents. Furthermore, although the correct source links were often provided, the current implementation lacks consistency in this regard. Due to the absence of a query classification mechanism, the chatbot always displays the top-ranked source link, even when it is not contextually appropriate. While this behaviour does not typically impair the user experience, it was addressed by both users and IT staff that, in cases where the chatbot was unable to provide a meaningful response, the accompanying source link also appeared irrelevant. This observation highlights the need for improved source attribution logic, ideally by integrating query classification to ensure that links are shown only when they are genuinely useful in supporting the provided answer.

With the evaluation and discussion of the results it is possible to reflect if the design objectives from the beginning were met. Table 7 provides an overview of the fulfilment status of the design objectives defined for the development of the RAG-based chatbot during treatment design. The evaluation reveals that most objectives were successfully achieved. Web-based access (O1) was implemented via a Streamlit interface, allowing intuitive interaction with the chatbot. The use of Kubernetes ensured a scalable deployment (O3), while the modular and model-agnostic architecture (O4) enables integration with both internal and external LLMs. The retrieval pipeline (O2) mostly succeeded in loading relevant documents from the CIT website, though gaps were identified due to missing subpages in the vector store, indicating room for improvement. Chat history persistence (O5) is only maintained within the current session, which limits its utility in long-term interactions. Nonetheless, the architecture is designed to support future extensibility (O6), allowing for the integration of additional data sources (e.g. other webpages of the university, pdfs, images) and functionalities (e.g. query classification, query rewriting, embedding of the webpage via streamlit, etc.). Overall, the technical implementation addressed the key design requirements, though certain features, particularly persistent memory and complete data coverage, require further development.

Table 8 presents the results of the validation of the expected effects defined in the problem context. The deployment of the RAG-based chatbot successfully delivered on the majority of user, operational, and economic expectations. From a user perspective, the chatbot met core criteria such as instant access to support, 24/7 availability, reduced frustration through conversational interfaces, and multilingual

ID	Design Objective	Achieved?
O1	Enable easy and web-based access to the chatbot.	✓ Achieved via Streamlit interface.
O2	Ensure accurate information retrieval by loading the original sources of the CIT website.	✓ Partially achieved; missing pages discovered.
O3	Ensure a scalable deployment capable of handling thousands of simultaneous users.	✓ Achieved through Kubernetes deployment.
O4	Use a modular software architecture that supports the integration of both internal and external LLMs.	✓ Achieved; model-agnostic setup implemented.
O5	Maintain conversational context through persistent chat history.	– Partially achieved; only session-based.
O6	Allow future extensibility for additional data sources and potential chatbot features.	✓ Achieved; extensible design supports this.

Table 7 Assessment of Design Objectives

capabilities via the underlying LLM. Although it maintains context during a session, the lack of persistent memory limits deeper context awareness across multiple interactions. Notably, this was the only partially met objective among the user-focused effects. In contrast, the expected effects for IT support staff remain unvalidated, as the chatbot has not yet been integrated into daily operations. While the survey with IT staff provided positive feedback on the chatbot's potential to reduce repetitive tasks and redirect human resources, these benefits could not be empirically confirmed. Effects such as reduced ticket volume, improved resource allocation, or increased job satisfaction must be observed over time in a live environment. Consequently, these outcomes are currently considered theoretical and should be re-evaluated in the context of a broader deployment. A long-term field study is recommended to verify the operational and economic impact of the chatbot in real-world support workflows.

Based on the discussion plenty of limitations for the implementation and evaluation emerges for this project which are presented in the following: Several limitations of the current implementation must be acknowledged. The chatbot's knowledge base was restricted to documents retrieved from the CIT website. As a result, questions referring to broader university services (e.g. Sciebo or the research portal) could not be answered reliably, leading to hallucinated responses or delayed fallback replies such as "I do not know." Furthermore, although the system is designed

for context-aware dialogue, chat history is only maintained within the current user session. There is no persistent memory that would allow users to revisit previous conversations across sessions or devices. Another limitation lies in the lack of support for multimodal input: users cannot upload screenshots or visual cues, which are often essential for describing technical issues. Additionally, the document retrieval pipeline revealed inconsistencies in scraping completeness. Some subpages (e.g. Remote Desktop installation instructions) were not captured due to either sitemap limitations or crawling logic flaws, resulting in incomplete or misleading answers. Lastly, the chatbot lacks query classification and relevance-based source ranking, occasionally presenting irrelevant sources when answers could not be confidently retrieved, an issue also observed by users and IT staff. On the other side, the evaluation of the chatbot was conducted in a pre-deployment phase with a limited set of test users and IT support staff, which presents several methodological limitations. Most notably, the expected organizational and economic effects, such as reduced workload, ticket volume, or support costs, could not be validated, as the chatbot has not yet been integrated into daily IT operations. Therefore, findings in this regard remain speculative and must be substantiated through longitudinal studies in a real-world setting. Furthermore, the evaluation relied primarily on survey responses, which reflect subjective perceptions rather than observable usage behaviour. While these insights are valuable, they may not accurately predict actual adoption or usage patterns. The scope of the performance evaluation was also limited to a predefined list of frequently asked questions. Although this ensured standardization, it excluded more complex, ambiguous, or error-prone scenarios, which are common in live support environments. Finally, without a formal baseline or control group (e.g. performance of a previous rule-based system), it is difficult to quantify the exact degree of improvement offered by the RAG-based approach. These limitations highlight the need for a broader and more integrated evaluation strategy in future work.

Based on these limitations recommendation for action can be made. First, the knowledge base should be expanded to include other relevant university platforms and services, such as the Sciebo documentation. This would significantly increase the chatbot's utility and reduce its susceptibility to hallucinations in queries beyond the CIT website's scope. Additionally, improvements in the scraping logic are necessary to ensure the completeness of the data source coverage, particularly by ensuring that nested subpages are correctly indexed and embedded. To support multilingual users more effectively, a revised onboarding message should be implemented to clearly communicate the chatbot's language capabilities and scope. The embedding of the chatbot into the CIT website would also increase the adoption. Further, pointing

out the capabilities of the chatbot and disclaiming offical advice. Furthermore, enhancing the prompting strategy to encourage counter questions in ambiguous cases and implementing query classification could help reduce misinterpretations and increase the reliability of responses. Lastly, for future evaluations, the chatbot should be deployed in daily operations, enabling data-driven analysis of user interaction patterns, support volume reduction, and cost savings over time.

Concluding with the developed artifact of the RAG-based chatbot for the IT Support of the University of Münster. While the chatbot developed in this study was tailored specifically for the IT Support context of the University of Münster, its architecture and implementation approach are highly transferable to other domains. The modular design, comprising a document ingestion pipeline, vector store, and model-agnostic generation layer, allows for easy adaptation to new knowledge domains by replacing or augmenting the source data. This means the chatbot could be repurposed for other university departments (e.g. library services, student administration) or even external institutions that rely on structured and unstructured documentation for user support. Additionally, because the LangChain framework supports integration with both internal and external LLMs and allows for domain-specific prompting, it is suited for use in highly regulated or knowledge-intensive environments such as healthcare, finance, or public administration. The ability to deliver contextual, source-grounded responses while remaining scalable and extensible suggests strong potential for generalizability, provided that the domain-specific documentation is made accessible and the prompting is adjusted accordingly. Thus, the artifact serves not only as a tailored solution for IT support but also as a blueprint for implementing reliable and maintainable RAG-based chatbot systems in other contexts.

Expected Effect	**Achieved?**
Instant access to support	✓ Chatbot provides immediate responses without delay.
Availability outside working hours	✓ Accessible 24/7 as a web-based application.
Reduced frustration	✓ Eliminates need to search through documentation; conversational input preferred.
Context awareness	– Partially achieved; context maintained within sessions but no long-term memory.
Multilingual support	✓ Translation via LLM works, though not clearly communicated to users.
Potential barriers (e.g., trust or sensitivity)	✓ Users prefer humans for complex or personal issues; boundaries respected.
Reduced workload	– Still to be validated; chatbot not yet integrated into daily operations.
Lower ticket volume	– Expected effect; future studies must validate in production use.
Improved resource allocation	– Not yet observable; depends on long-term integration.
Fewer interruptions	– Not measurable until broader deployment.
Support for onboarding	– Potential use case, not evaluated in this study.
Increased job satisfaction	– Remains to be assessed after integration.
Reduced support costs	– Economic impact must be shown in practice.
Lower operational overhead	– Theoretical benefit; not yet quantifiable.
Scalable support	✓ Can handle increased demand during peak periods.
Long-term savings	– Still hypothetical until sustained deployment.

Table 8 Assessment of Expected Effects

7 Conclusion

This thesis set out to explore the potential of RAG for LLMs in the context of university IT support, specifically within the operational setting of the University of Münster. The underlying research problem addressed how a RAG-based chatbot could improve the IT Support by reducing the number of repetitive requests and by that reducing personnel cost or redirecting human personal to more valuable tasks and by increasing the availability of the IT service and by making the access to information for students and staff member easier. The motivation originates from previous limitations observed in rule-based chatbot systems, which were considered inadequate due to their inflexibility, high maintenance demands, and inability to handle open questions. With the rise of RAG for LLMs, a new opportunity emerged to implement a more dynamic, context-aware chatbot capable of addressing a broader range of user needs in a scalable and maintainable way. To approach this challenge, a comprehensive theoretical background study was conducted to understand the mechanisms of RAG for chatbots, its core components, namely retrieval and generation, and the recent advances in its application for domain-specific use cases. This literature review formed the conceptual foundation for the artifact's design and informed the use of DSR methodology. Within the DSR process, the chatbot was defined as the artifact and developed in close alignment with the stakeholder needs identified through discussions with the Head of IT Services. A set of six design objectives was formulated, including web accessibility, accurate information retrieval from official CIT sources, scalable deployment, modular architecture, session-based context awareness, and extensibility for future use cases.

The implementation followed a modular system architecture, utilizing LangChain as the core RAG framework, gpt-4o-mini as the external LLM, and ChromaDB as the vector database. All documents from the official CIT website were scraped and embedded to serve as the basis for document retrieval. The frontend was implemented using Streamlit, ensuring web-based accessibility, and the entire application was deployed on the university's Kubernetes cluster to ensure scalability and availability. To evaluate the artifact, a multi-layered methodology was established, including three evaluations: a user study with students and staff, a targeted IT support staff study, and a factual performance test covering frequently asked support questions. Each of these evaluations focused on different aspects of the system, allowing for both technical and experiential insights.

The user study revealed that the RAG-based chatbot was widely perceived as a faster and more convenient tool for finding IT-related information than traditional

methods such as web search, the CIT website, or general LLMs. While users acknowledged the reliability of the CIT website, they consistently ranked the chatbot highest in terms of speed and ease of use. Notably, users expressed a clear preference for using the chatbot as a first point of contact, particularly for straightforward IT issues, and would only escalate to human support when necessary. This finding suggests that the chatbot could effectively function as an initial support layer, reducing unnecessary load on IT personnel. The chatbot's ability to clarify ambiguous queries, maintain conversational context, and retrieve answers with sources also contributed to increased user trust, although some expressed a need for more transparency about the chatbot's capabilities and limitations. The staff study yielded complementary findings. IT support personnel recognized the chatbot's utility in addressing standard and repetitive support requests. While they did not consider it a full replacement for the IT hotline, they did see its potential to free human resources for more complex or system-specific issues, ultimately supporting a more efficient allocation of work. From an economic perspective, the staff also noted possible gains through 24/7 service availability and reduced ticket volumes, though some remained cautious about whether these effects would scale in practice without broader deployment. The performance evaluation further validated the chatbot's capabilities. Out of eight frequently asked questions, five were answered with full accuracy, completeness, clarity, and correct source attribution. In two cases, partial points were deducted due to insufficient clarification or missing steps in the explanation, yet the responses still demonstrated contextual awareness and adaptability. One question, related to remote desktop access, revealed a critical weakness in the scraping logic, where certain subpages of the CIT website were not included in the vector store. This incident highlighted the importance of thorough preprocessing, source crawling, and fallback handling within a RAG system. The findings from this evaluation not only confirmed the chatbot's capability to handle real-world IT support scenarios but also emphasized the need for continuous monitoring, improvement of prompt engineering, and enhanced retrieval logic.

The discussion of results led to several important implications. On the one hand, it became evident that the current implementation largely fulfilled the defined design objectives. The chatbot was accessible, scalable, and accurate in retrieving content from its document base, while allowing for future extensibility and model integration. However, limitations also came up, particularly with respect to persistent chat history, inconsistent source attribution, and lack of multimodal input. Suggestions such as the integration of login-based chat memory and support for image uploads (e.g. error screenshots) were derived from both user and staff feedback. Furthermore, the findings underscore the importance of clearly communicating the

chatbot's capabilities and limitations to users, particularly its language support and domain-specific scope.

In conclusion, the RAG-based chatbot developed in this thesis demonstrates a promising solution to the challenges faced by university IT support services. It fulfills its role as a 1st-level support system by providing accurate, fast, and convenient answers to common queries while relieving pressure on human staff. The chatbot can serve as a valuable assistant in operational environments where standard information needs are frequent and documented. In terms of the research problem, the results affirm that RAG-based chatbots, when customized for a specific knowledge base and context, can meaningfully enhance institutional support services. Within the current research landscape, the developed system represents a state-of-the-art, query-based application of RAG that aligns with the latest advancements in LLM technologies. As such, it serves not only as a practical tool for the University of Münster but also as a blueprint for similar implementations in other organizational contexts. Looking ahead, future research on the exploration of RAG should focus on the integration and evaluation of RAG enhancements. For the continued development of the RAG-based chatbot within the university's IT support, particular focus should be placed on incorporating multimodal input, refining prompt engineering techniques, and extending the scope of document retrieval to improve both reliability and user experience. Finally, a large scale productive deployment of the chatbot should be pursued to enable a more accurate assessment of its economic impact and operational value in a real setting for the IT support.

Appendix

A User Study Questionnaire

IT Support Chatbot Evaluation - User Study

As part of this master's thesis, a chatbot for the IT support at the University of Münster was developed. The purpose of this questionnaire is to evaluate the newly developed chatbot.

This questionnaire consists of two parts:

- The first part evaluates different approaches to find an answer to IT-related questions.
- The second part focuses on a detailed evaluation of the IT Support Chatbot.

The entire questionnaire will take approximately 15–20 minutes to complete.

You can access the new chatbot via the following link: https://rag-thesis.uni-muenster.de/

You can answer open questions either in English or German.

Part I:

Here are four frequently asked IT-related questions. Your task is to find answers using different approaches.

Questions:

(1) I have forgotten my password. What do I have to do now?

(2) How can I extend my university ID?

(3) How do I set up an OTP generator?

(4) How do I establish a VPN connection?

Approaches:

(1) Perform a web search (e.g., Google, Bing, Yahoo) to find an answer.

(2) Use a Large Language Model (ChatGPT, UniGPT, Claude, Gemini, etc.) of your choice to find an answer.

(3) Use the CIT Website (https://www.uni-muenster.de/IT/services/index.html) where you can find the answers to the questions.

(4) Use the newly developed IT Support Chatbot (https://rag-thesis.uni-muenster.de/) to find an answer.

You can decide which question to answer with which approach. However, please ensure that each approach is used at least once.

Once you have completed this task, please answer the following questions below:

(1) **Which approaches were able to provide a correct answer to your question?**

- Web Search ☐
- LLM ☐
- CIT Website ☐
- IT Support Chatbot ☐

(2) **Rank the approaches from the most convenient way (1) to the least convenient way (4) to find an answer to a question.**

Approach	**Rank (1–4)**
Web Search	
LLM	
CIT Website	
IT Support Chatbot	

(3) **Rank the approaches from the fastest (1) to the slowest (4) in terms of response time for finding an answer to a question.**

Approach	**Rank (1–4)**
Web Search	
LLM	
CIT Website	
IT Support Chatbot	

(4) **Rank the approaches based on the quality and accuracy of the answer to your question, from the most detailed and correct (1) to the least detailed and correct (4).**

Approach	**Rank (1–4)**
Web Search	
LLM	
CIT Website	
IT Support Chatbot	

(5) **Please elaborate on how you would search for an answer to an IT-related issue in the context of the University.**

(6) **What were the biggest challenges and advantages of each approach? You can answer in bullet points.**

Part II

In this part, you are invited to try out the new IT Support Chatbot in more detail.

Link: https://rag-thesis.uni-muenster.de/

Here are some more frequently asked questions for example to ask:

- How do I establish a connection to the Remote Desktop?
- How do I get Office 365?
- How do I set up my WLAN?
- I can't find the poster printer. Why is that?
- My OTP generator no longer works. What do I have to do now?

If you are not familiar with IT-related questions, you can simulate being a new student at the University of Münster seeking onboarding information.

Feel free to be creative and come up with your own questions.

Please take around 5 minutes to try out the new chatbot, then fill out the questionnaire below.

(a) **Please indicate your level of agreement with the following statements.**

Statement	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
The chatbot's responses were complete and accurate to answer my questions.	☐	☐	☐	☐	☐
Using the chatbot to find my answer felt quick.	☐	☐	☐	☐	☐
I felt that the chatbot was able to retain context and remember previous messages throughout the conversation.	☐	☐	☐	☐	☐
I believed the chatbot understood and responded to me like a human would when answering questions.	☐	☐	☐	☐	☐
The chatbot's conversational style influenced my willingness to use it in the future.	☐	☐	☐	☐	☐
The chatbot allowed me to resolve my IT issue with less effort compared to web search, LLMs, or the CIT website.	☐	☐	☐	☐	☐
From now on, I would prefer the chatbot over other ways to obtain information on my IT-related problems.	☐	☐	☐	☐	☐
It was easy for me to use the chatbot to get my question answered.	☐	☐	☐	☐	☐
The visual appearance of the chatbot made it easier for me to interact with it.	☐	☐	☐	☐	☐
The chatbot's design and personality made it more engaging to interact with.	☐	☐	☐	☐	☐
I was satisfied with the overall experience of using the chatbot.	☐	☐	☐	☐	☐

(b) **What do you like about the chatbot?**

(c) **What do you not like about the chatbot?**

(d) **Would you use the chatbot instead of searching on your own? Why or why not?**
And when would you use the chatbot instead of searching on your own?

(e) **Would you use the chatbot instead of human support? Why or why not?**
And when would you use the chatbot instead of human support?

(f) **Do you have suggestions for improvement? Are there crucial parts missing? Did you encounter major issues?**

B IT Support Staff Study

IT Support Chatbot Evaluation – IT Support Team Study
As part of my master's thesis, I developed a chatbot to support the IT services at the University of Münster.
This questionnaire aims to evaluate the chatbot's effectiveness and usability. Completing the survey will take approximately 15 minutes.

The chatbot leverages the LLM GPT-4o-mini and retrieval-augmented generation (RAG) to access and provide information from the CIT website.

You can access the chatbot via the following link: https://rag-thesis.uni-muenster.de/
Please note that the website may display a security warning. You can ignore this message and proceed with the evaluation.

To participate, please test the chatbot's ability to function as a first-level IT support assistant by asking IT-related questions.

Afterwards, kindly complete the questionnaire below. You can answer open questions either in English or German. Thank you for your time and support!

(1) **Please indicate your level of agreement with the following statements.**

Statement	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
The chatbot is able to provide complete and accurate responses to IT-related questions.	☐	☐	☐	☐	☐
The chatbot was able to retain context and remember previous messages throughout the conversation.	☐	☐	☐	☐	☐
The chatbot understood and responded like a human would when answering questions.	☐	☐	☐	☐	☐
The chatbot's conversational style is similar to a real IT support person.	☐	☐	☐	☐	☐
The chatbot is capable to resolve IT issues like the human support personnel.	☐	☐	☐	☐	☐
I believe that the chatbot can be part of the 1st-Level Support from now on.	☐	☐	☐	☐	☐
I believe that it could relieve or compensate for personnel shortages.	☐	☐	☐	☐	☐
I believe the chatbot will have a huge adoption of users.	☐	☐	☐	☐	☐
The chatbot's design and personality made it more engaging to interact with.	☐	☐	☐	☐	☐
The chatbot could be a beneficial way to increase the availability of IT Support.	☐	☐	☐	☐	☐
The chatbot provides an easier way to find sources on the CIT website than searching by yourself.	☐	☐	☐	☐	☐
Do you think the chatbot can help reduce the number of repeated requests of frequently asked questions?	☐	☐	☐	☐	☐
Do you think the chatbot is able to completely substitute the Support Hotline?	☐	☐	☐	☐	☐

(2) **What do you like about the chatbot?**

(3) **What do you not like about the chatbot?**

(4) **Which support requests do you think could be fully resolved by the chatbot?**
And are there certain types of requests that the chatbot cannot answer and still require human assistance?

(5) **What kind of improvements (e.g. cost-savings, redirection of valuable human resources to other important tasks, increased availability of the service, overall higher customer satisfaction, etc.) do you expect when the chatbot is part of the 1st-Level support and why?**

(6) **Do you have suggestions for improvement? Are there crucial features missing? Did you encounter major issues or limitations?**

C Content Analysis of Open Questions

C.1 User Study:

Question 5: Please elaborate on how you would search for an answer to an IT-related issue in the context of the University.

ID	Category			Response
1	G	W	C	before I would have probably used Google and then go via the CIT website; but now I would use the IT Chatbot because it was the most detailed, understandable and easiest way to find an answer
2	W	C		First I would visit the university's IT website. If I see that there is a Support Chatbot available like in this survey I would certainly use it because it can help me the fastest.
3	G	P		Usually through a google search or communication channels
4	L	C		LLM or IT Support Chatbot
5	C	W	L	Falls der IT Support Chatbot schon verfügbar ist, würde ich ihn benutzen [...] Ansonsten würde ich erstmal auf der CIT Website schauen und wenn ich dort nichts finde, Chat GPT fragen.
6	C	W		Der IT Chatbot wirkt am schnellsten und passend detailliert für IT Fragen. Vor allem im Gegensatz zum CiT
7	G			I would do a google search
8	G	W		Google the problem, which normally leads me to the desired part of the CIT website.
9	G	W		Google and look für university held pages
10	G	W		googlen, dann kommt man entweder auf die CIT-Seite oder eine passende andere Webseite.
11	G			I would google some keywords of my problem and hope that it would give me the relating websites.
12	G	C		I would google it first because it's very convenient [...]. If I am not forwarded [...] I would ask the IT Support Chatbot
13	P	W	C	normally I ask my colleagues or use the search on the CIT Website, but I will try the IT support chatbot
14	W			CIT website
15	G	W		Usually using a search engine to find the right "entry point" on the websites
16	W	C		I have always used the CIT website until now, but the chatbot would be a good alternative
17	G	W		My first step is always to google and most times I am provided the desired link to a University Page

Table 9 Categorization of IT Support Search Approaches - Part 1

ID	Category	Response
18	G W C	Without the IT Support Chatbot: google the topic + university münster [...] subpage of the CIT homepage [...] With IT Support Chatbot [...]
19	G C	Currently, usually via web search. But if reliable, I'd always prefer a Chatbot
20	W	Probably would check the website first
21	G C	I would try a web search first [...]. Though a fully functional IT Support Chatbot would be even faster
22		Be sure that it comes from an official source.
23	W C G P	First I would check the CIT service portal [...]. Now I would try the solution as described. If this does not work, I would either google or ask colleagues or write to the CIT support.
24	G	google
25	W G	CIT Website or Web Search

Table 10 Categorization of IT Support Search Approaches - Part 2

C.2 User Study: Question 6

Question 6: What were the biggest challenges and advantages of each approach. You can answer in bullet points.

ID	Category	Response
1	Speed, Accuracy, Ease of Use, Visual Design, Completeness, Reliability	Web Search: + fast + no extra website I have to use - gave the most inconvenient answer - did not give all options available just one; LLM: + easy + fast detailed answer + can explain further if needed - did not give all available options to solve the problem - did not give links; CIT Website: + gave all options available - complicated to use - no clear, easy or fast overview - don't know where to look for what - no nice design; IT Chatbot: + easy to use + provides all available options + clear structure + answers fast and understandable + nice design + gives hints what to keep in mind and take care of when using the options (like that the password reset can take some time) + can explain further if needed - gives security warning - can only use it via the link, not directly on the website
2	Navigational Challenges	The challenge most of the time is, that in an university environment there are many different departments and sometimes it makes it difficult to find the information you need.
3	Speed, Accuracy	Advantage of the Chatbot: accuracy and speed. Challenges for the others were either the speed or the accuracy.
4	Speed	Fast answers.

Table 11 Categorization of Challenges and Advantages – Part 1

ID	Category	Response
5	Navigational Challenges, Speed, Completeness, Human Support Availability	Websuche: Man muss sich durch viele Seiten durchklicken und sucht lange nach der passenden Antwort. Chat GPT oder Ähnliches: Funktioniert ähnlich gut wie der IT-Chatbot und verweist bei Angabe der Uni auch auf passende Websites. IT Support Chatbot: Liefert zumindest auf die 4 Beispielfragen sehr ausführliche Infos zur direkten Problembehebung. CIT Website: Man muss sich etwas länger durch alle Reiter klicken aber auch hier findet man eigentlich gute Hilfe. Außerdem hat man einen Überblick über alle Bereiche und findet Kontaktinformationen, falls man alleine nicht weiterkommt – die erhält man aber auch (spätestens auf Nachfrage) vom IT Support Chatbot. Generell liefert der Chatbot am schnellsten die gewünschte Information.
6	Navigational Challenges, Accuracy, Completeness	Bei der Websuche muss man selbst Seiten finden wo die Antwort steht, sie ist nicht direkt parat. Cit lieferte kein passendes Ergebnis nur Artikel wo Worte der Frage vorkamen. ChatGPT gab direkt eine Antwort allerdings allgemein. Der Chatbot gab eine allgemeine Antwort und konkretisierte direkt und fragte nach einem weiteren Detail woraufhin direkt eine ausführliche Schritt-für-Schritt Anleitung folgte.
7	Accuracy, Navigational Challenges, Completeness	LLM: is sometimes wrong and presents a solution to a slightly different problem. Websearch: I have to look through 1–3 websites. Chatbot: I cannot intuitively see further helpful links that I would see on a website.
8	Navigational Challenges, Accuracy, Detail Level	The CIT website is difficult to navigate (to which category belongs OTP? where can I find information on how to prolong my group memberships?) – the web search usually navigates me to the right part of the CIT website – the LLM (uniGPT) takes quite long and is very vague, therefore not a good option – I really like that the IT Support Chat Bot also provides me with a website, where I can check if the answer given was correct. Overall, the level of detail the Chat Bot provides is much higher than with the other options.

Table 12 Categorization of Challenges and Advantages – Part 2

ID	Category	Response
9	Speed, Accuracy, Navigational Challenges, Trust	Chat bot: fastest and easiest, but I'm scared it's wrong. CIT website: I get the right answer but the menu is not good. Google: convenient and faster than the website because I get to the right subpage directly. LLM: I know what page to go to, but the answer is wrong.
10	Accuracy, Specificity	Googlen liefert das richtige Ergebnis in den meisten Fällen – Webseite hat die wirklich korrekte Anleitung mit passenden uni-spezifischen Parametern.
11	Speed, Specificity, Usability	- Using the google search and the website itself needs much time (a lot of text to read). - Using the IT chatbot and chatGPT is fast. - The IT chatbot gives me specific information about our university, not of the whole world.
12	Accuracy, Complexity, Presentation	- LLM: Information given was partially wrong – so I had to consider whether the information given was right or wrong. - IT Support Chatbot: experience was fully positive. - Web search: Very effective by using bullet points only but is not appropriate for more complex questions. - CIT website: same as web search.
13	Relevance, Navigation	Web Search and LLM: not university specific. CIT-Website: had to search first. IT Support Chatbot: none.
14	Chatbot Tone, Target Group	If you know the CIT website a little bit, it's just OK to use it. The Chatbot is quite chatty if you know what you want, but I guess it's a very good option/addition for people who are new to this, especially new students.

Table 13 Categorization of Challenges and Advantages – Part 3

ID	Category	Response
15	Information Currency, Navigation, Speed, Accuracy	* CIT Website: Most up-to-date information; navigation to the correct sub-page can be tricky. * Web Search: Extremely fast result for the general direction; manual review of result page necessary. * Support Chatbot: Quite up-to-date information and good details (e.g. password differences); longer waiting time for answer. * LLM: Inaccurate/outdated information; longer waiting time; ease of use.
16	Personalization, Media Integration	- I really like the screenshots included on the CIT Website. Otherwise the IT support Chatbot and the LLM provided similarly good results. - The chatbot is focused on my specific problem, on the website I have to filter for my cases (android, ios,...). - It would be amazing when I could send screenshots with error messages to the chatbot and the screenshots from the website are included in the bot.
17	Trust, Accuracy, Redundancy	The general LLM is no option, as it only provides general Information. However, all of the other three methods only differ in how to reach the university page, with the addition that the Chatbot already provides detailed step-by-step instructions. [...] Furthermore, my slight distrust in LLMs makes me choose the Webpage if provided the option. Lastly, the webpage, providing pictures is nice, even though they might be outdated.
18	Accuracy, Reliability, Information Structure	Random LLM: does not give accurate information / does not represent the answer/solutions my employer wants me to use for IT-related things. Google: sometimes links to old/obsolete CIT pages or "obscure" subpages. CIT webpages: good if I do not know what I am looking for. Chatbot: combines speed and precision of Google with CIT webpage advantages.

Table 14 Categorization of Challenges and Advantages – Part 4

ID	Category	Response
19	Structure, Reliability, Overview	CIT Website: (-) understanding structure and finding info; (+) screenshots, broader service view. Web search: (-) choosing correct site; (+) fast and complete info. LLM: (-) reliability of sources; (+) quick overview. Chatbot: (+) fast, clear, with source references.
20	Search Strategy	Coming up with a search strategy.
21	Navigation, Query Formulation, Specificity, Convenience	- Biggest challenges: navigating the CIT website, formulating the right search term for the web search, LLM giving very general responses that don't actually apply to this specific situation (e.g., advice on what to do after forgetting my password). - Biggest advantages: the high accuracy of the web search and IT Support Chatbot, the Chatbot being more convenient (no need to click and read through web page, provides an accurate summary).
22	Consistency, Specificity	The Chatbot was super convenient – but its answers did not always match the CIT website. For example lost password: CIT section shows different info than the site the chatbot offers. Web search and LLM are too general. I would need more specific information.
23	Structure, Detail Level, Trust	IT support chatbot: Detailed, OS-specific branches asked directly, good feeling (being guided). Google: worked well, led directly to the correct page. LLM: too general, unsure about correctness, no links. CIT website: everything fine, but had to search the CIT URL via Google.
24	Navigation	CIT Website: Sometimes difficult to find the topic I'm looking for.

Table 15 Categorization of Challenges and Advantages – Part 5

Survey Statement	**Assigned Dimension**
The chatbot's responses were complete and accurate to answer my questions.	Quality and Accuracy
Using the chatbot to find my answer felt quick.	Efficiency
I felt that the chatbot was able to retain context and remember previous messages throughout the conversation.	Conversational Memory
I believed the chatbot understood and responded to me like a human would when answering questions.	Natural Language Processing (NLP)
The chatbot's conversational style influenced my willingness to use it in the future.	Anthropomorphism
The chatbot allowed me to resolve my IT issue with less effort compared to web search, LLMs, or the CIT website.	Usability
From now on, I would prefer the chatbot over other ways to obtain information on my IT-related problems.	User Experience (UX)
It was easy for me to use the chatbot to get my question answered.	Usability
The visual appearance of the chatbot made it easier for me to interact with it.	User Interface (UI)
The chatbot's design and personality made it more engaging to interact with.	Personas
I was satisfied with the overall experience of using the chatbot.	User Experience (UX)

Table 16 Categorization of Likert scale Questions into Evaluation Dimension Metrics

ID	Category	Response
1	Response Quality, Step-by-Step, Human-Like	That it answers like a human being and can even specify if I didn't understand something immediately and needed a more deep and detailed explanation and that it provides links and gives godly structured answers I can follow step by step.
2	Structured Responses	The structured answers. It gave enough information but not too much information regarding the question at hand.
3	Speed	How fast it can answer my questions.
4	Speed, Detail, Clarification, Design	Man erhält schnell Hilfe, der Chatbot ist sehr ausführlich und kann einzelne Dinge auf Nachfrage nochmal genauer erklären. Er ist optisch ansprechend.
5	Accuracy, Clarification, Step-by-Step	Findet richtige Antworten auch mit Schreibfehlern in der Frage. Wenn die Frage zu kurz ist, wird nachgefragt. Detaillierte Schritt für Schritt Antwort auf Frage.
6	Step-by-Step, Embedded Links	You always have stepwise instructions links directly embedded in the answers.
7	Source Referencing, Clarification	It always states the sources of information it uses so that I can verify the answers. Also, it asks for further details to provide a better answer. So in general, it offers the information from the CIT website, but with better navigation.
8	Convenience, Source Referencing	It's convenient and fast, there is a link for fact checking.
9	Accuracy	Gibt korrekte – wenn auch teils allgemeine – Antworten aus.
10	Friendliness, Speed	It is very friendly and it is fast.
11	Source Referencing, Accuracy	Quelle wird direkt mit angegeben – richtige Antworten auch auf komplexere Fragen.
12	Human-Like, Source Referencing	It felt like asking a colleague but being provided the necessary links at the same time.
13	Onboarding	Great start for desoriented newbies.
14	UI, Tone	Simple interface, "friendly" responses.
15	Convenience	It is quick and less annoying. I do not have to provide more context like the University of Münster.
16	Speed, Accuracy	Good response time and seems accurate.

Table 17 User Feedback on Liked Aspects of the Chatbot - Part 1

ID	Category	Response
17	Source Referencing	That it directly links to sources and gives the link at the bottom of each answer.
18	Speed, Design, Source Referencing, Clarification, Multilingualism	Speed, reduced design, highlighting relevant aspects, sources, available languages, unlike other LLMs it directly asked for details needed to filter and substantiate the information (e.g. operating system).
19	Convenience	Convenience.
20	Usability, Speed, Surprise Features	Easy to use, provided download link for WinAuth which positively surprised me, quick and concise.
21	Convenience, Accessibility	It is very convenient and quick. Especially for non-IT users it might feel more natural than contacting a human being and waiting for a more or less nice / supportive answer. [...] I like the running icon :)
22	Simplicity, Accuracy	Einfach, scheint auf die richtigen Informationen Zugriff zu haben. Kurze, prägnante, ausreichend ausführliche Antworten.
23	Usability	Very easy to use.

Table 18 User Feedback on Liked Aspects of the Chatbot - Part 2

ID	Category	Response
1	Memory	It doesn't remember the conversation when I close and then use the link again. And that it gives a security warning in the beginning.
2	Design	Maybe the design could be a bit more original. But I find it difficult to find something I disliked about the chatbot.
3	Nothing	Everything was good.
4	Verbosity	Manchmal vielleicht etwas "zu" ausführlich bei einer simplen Frage, aber das liegt vermutlich an der Frage und am Vorwissen der fragenden Person. Mir ist nichts Negatives beim Ausprobieren aufgefallen.
5	Typing Speed	Man muss warten bis der Chatbot alles ausgeschrieben hat. Man ist gewohnt, dass die Nachricht direkt komplett aufploppt.
6	Incorrect Answer / Domain Limitation	I asked for a non-IT related question (how to obtain new student ID card) [...] maybe this can be handled better.
7	Incorrect Answer	When I prompted that I had forgotten my password, it stated that I could go to the IT portal to reset it – uniGPT at least recognized that this was not a valid option.
8	Trust	I have to remember a new website, I don't trust it, cause ChatGPT is always wrong, so I have trust issues with LLMs.
9	Specificity / Typing	Diese "Wort für Wort"-Schreiben. Ergebnisse nicht sehr spezifisch und allgemein gehalten mit vielen Optionen, die nicht relevant sind.
10	Personalization	That it has no name :) That the jokes don't make sense.
11	Nothing	Nothing.
12	Conversational Style	As always, if you know what you want, chatbots are a bit too chatty for me.
13	Hallucination / Privacy	* Hallucinations with more specific questions [...] * No warning about data being sent to an external service.
14	Incomplete Answers	Sometimes it forgets to mention details and gives incomplete responses.
15	Irrelevant Suggestions	If it cannot answer it still provides links to unrelated webpages sometimes.

Table 19 User Feedback on Disliked Aspects of the Chatbot - Part 1

ID	Category	Response
16	Speed / Tone / Customization	That it is set up to "write" very slowly. That it is set up to use smileys as a default. That it is set up to "act" like a conversational human.
17	Incomplete / Incorrect Answer	Some answers were incomplete (which the bot itself noted). Reference was made to the source, but it was incorrect.
18	Incorrect Answer	Sometimes gave incorrect answers (I asked how to install a certificate in my Outlook).
19	UI Placement	Takes up the whole screen. I would rather prefer it as an icon on any university website.
20	Typing Animation	I am not a big fan of typing animations (in general, not specific to the chatbot) and prefer answers "on one glance".
21	General Rejection of AI	Künstliche Intelligenz verschwendet Ressourcen. Eine Websuche bringt mich auf das gleiche Ergebnis, ich muss nur ein bisschen weniger faul sein.

Table 20 User Feedback on Disliked Aspects of the Chatbot - Part 2

ID	Response
1	I would use it if I need a description on how to do anything IT related, especially when I need step-by-step descriptions. Because Google and the CIT website are not well structured and it´s way faster and easier to use the Chatbot.
2	Yes i would use it because if prompted correctly it finds exactly the information i need in a structured overview.
3	Yes! Because its faster and accurate
4	Gerade bei komplizierteren Fragen würde ich den Chatbot benutzen, weil er alles sehr verständlich erklärt.
5	Für konkrete Anleitungsfragen v.a IT wo man im Web nicht direkt die richtige Lösung findet ohne sich durch tausend websiten durchzusuchen
6	I've never had an IT problem so I dont know. I would probably do a web search first, as i have google opened faster than the chatbot. I would have to search for the link to the chatbot first. But when I dont find anything in one or two web searches, I would look up the bot.
7	I think this would offer a great entry point on where to find the solution to a problem.
8	Yes, faster and I can ask context questions without having to do a new search
9	Nein, die Webseite mit Bildern und Unterkategorien der Uni (welche meistens über die Websuche gefunden wird) ist intuitiver zu verstehen.
10	I would use the chatbot for simple information about something. Not for information where small details are necessary, e.g. the manuals for installing specific software, where little details during the installation make the difference between success and failure. And sometimes it is helpful to see pictures of the installation way.
11	Wenn ich mich in die Lage eines normalen Nutzenden reinversetzte auf jeden Fall, da vor allem komplexere Probleme eher schwierig zu lösen sind, wenn man sich nicht gut auf der Seite des CIT auskennt
12	it feels a bit too convenient and sometimes I like to challenge myself and find things out myself
13	i would use the bot for topics which I am less sure about
14	Probably would use it from time to time for more specific questions where a simple web search would not suffice
15	It is easier and faster. I would need a search string for google, then check the results, find the best tab, then read the page until I find what I need. For the bot I only need a search string or prompt.
16	Probably not, because I would need to search for the Chatbot instead of searching directly first. I would probably use it if it is embedded in the University pages, so that if i do not find the Information that I am looking for, i can ask the chatbot

Table 21 Responses to Question 10: Would you use the chatbot instead of searching on your own? - Part 1

ID	Response
17	If the information given by the chat bot can be trusted to be correct, i.e. it is equally official to information on the CIT homepage, I would use it. If the answers of the chatbot are nor regarded as officially correct, I see no point in using it, if I have to verify by hand that the answers are in accordance with my employers regulation anyway.
18	If reliable, yes, for all questions related to service information (troubleshooting, overviews, manuals). Simply because it's much more convenient.
19	Yes, it is convenient
20	I would use the chatbot instead of searching on my own when I need 'basic' questions answered. For more complex issues, the solution to which isn't spelled out in a database or on the CIT website, I prefer to call the IT support.
21	for anything which might is crucial for security I would like the official source to be sure.
22	Probably yes - but I am not someone to uses search options as first step. Thus I would probably forget that the chatbot is a more convenient method :)
23	i would use the chatbot, because i think it might be a quicker than searching by myself

Table 22 Responses to Question 10: Would you use the chatbot instead of searching on your own? - Part 2

ID	Response
1	yes I would, because I can talk to it like a human and ask as many times as I want to without it getting annoyed or think I am stupid. And I could use it whenever I want and not only during service hours.
2	I think there are still somethings which are better solved with a human because the interaction can be faster. But for getting information a human is in the disadvantage in my opinion.
3	For small questions I would use the chatbot. For bigger once I would want human support
4	Bei einigen Dingen würde ich lieber persönlichen Support erhalten, da ich mir alles direkt am Laptop zeigen lassen kann, das hilft manchmal finde ich. Aber ansonsten würde ich den Chatbot vorziehen, weil er schneller antwortet. Bei einem Anruf/ persönlichen Termin ist man auf Öffnungszeiten angewiesen, per Mail wartet man lange auf Antworten. Außerdem ist der Chatbot immer freundlich:)
5	Chatbot geht natürlich viel schneller als eine Person hinzuziehen, jemanden zu erreichen und dann den Sachverhalt zu erklären. Wenn man ein Problem selbst lösen möchte, würde ich den Chatbot bevorzugen Chatbot konnte auf meine Nachrichten eingehen, Antworten liefern, wenn ich nur ein Wort liefere wird nachgefragt, wenn ich Schreibfehler mache wird trotzdem die Frage richtig verstanden
6	I would use the chatbot for general problems. If it is really specific and no one else had this problem before, I would talk to a human
7	Chat bot can solve easy tasks that are already explained on the web site. More specific questions or more complex problems (even if they start with an easy question) still need to be solved by a human. I suppose it is also a good option for people that are afraid of writing an e-mail / calling the IT support directly.
8	Yes, faster, people are unfriendly and don't always talk on a "normal persons" level of it understanding
9	Ja, wenn es um einfache Sachverhalte geht.
10	I would use the chatbot for simple questions or information. Not for complex information where more relations to user groups eg. are important. For that specials I would contact the human support.
11	Ja würde, da es weniger aufwändig ist. Sollte es um personenbezogene Daten gehen, würde ich wahrscheinlich aber eher noch den menschlichen Support wählen.
12	I think the chatbot could be a good first point of contact but if I cannot find a solution having human support is definitely important.
13	Most likely yes, because it would always be immediately available and would not 'overlook' obviously existing information. As soon as I would believe I have a 'non-standard' problem, I would prefer human support.

Table 23 Responses to Question 11: Would you use the chatbot instead of human support? - Part 1

ID	Response
14	Yes, I can ask stupid questions, when I can angry I can insult it. My first try would always be the chatbot. As soon as I have problems the bot can't resolve within a few promts or causes new problems, I would ask for human support
15	yes, if i would not be able to find the information directly, i would first ask the chatbot before calling human support. Mostly because I feel not needing to bother them, if I can find out quickly myself with the chatbot
16	No, I would not use the chatbot instead of human support. If I ask for human support it is, because the CIT webpage tells me to contact IT/IVV regarding my issue or I do have a question which needs human approval (e.g. can we pay for this? is this not yet listed cloud platform ok to use according to the DSGVO?)
17	Would always use it before contacting human support. Often support is contacted, because information can't be found (quickly enough) although it's there.
18	gives flexibility and is good for shy people
19	For individual issues (such as my device not working properly, printer suddenly not connecting) I prefer human support since it allows me to explain my issue more clearly - and if I struggle explaining my issue, the person on the other end is able to help me out. As mentioned above, I would use the chatbot for more basic questions, the answer to which I could also find on my own on the CIT website or through a web search - using the chatbot would be much more convenient and time-efficient in these cases.
20	I would refer human support if they are friendly and fast (hotline) and for more complex qustions.
21	As someone who offers IT support: Many people will prefer a chatbot as it feels less "embarrassing" to ask a machine than asking a human. Even more so as some colleagues in IT support are not very good at dealing with insecurities and "noob questions" from their counterparts. The chatbot does not judge. :) For me, I would prefer a chatbot for easy problems like the (very good) examples in this survey. For more complex or very specific questions, I would prefer human support.
22	Bei Informationen, die so speziell sind, dass der Chatbot möglicherweise falsche Antworten geben würde.
23	the chatbot is very convenient, I don't like to make phonecalls and mails often have quite a long response time, so I would definitely try the chatbot first and only use the human support if I can't find a solution to my problem

Table 24 Responses to Question 11: Would you use the chatbot instead of human support? - Part 2

ID	Response
1	It would be nice if there would be previous conversations that I could look at again instead of once I close it I have to ask the same questions again and cannot just quickly go to the previous conversations.
2	No.
3	Everything was good
4	Mir fehlt nichts. Vielleicht könnte man zukünftig noch einrichten, dass der Chatbot auch Fragen auf spanisch beantworten kann für zB Ersamus-Studenten?
5	Besser wär wenn die Antwort direkt aufploppt. Aber das macht auch den Chatcharakter und dass die Antwort gerade kommt. Ansonsten vollkommen in Ordnung, sogar sehr zufriedenstellend
6	I didnt notice major issues
7	Sometimes I don't understand the source it gives to me. I asked why I cannot login to a very specific service (Beck-Online) and the source it prompted to me was the onboarding wizard. Since the answer was not helpful and the source was not either, this was overall not a good interaction.
8	Most information is correct, but I found one big mistake: The chatbot mixes up the IT portal and the intranet for students. The campus management system is not found in the IT portal, but in the intranet for students.
9	Ist es evtl möglich Auswahlmöglichkeiten direkt zu verlinken, anstatt sie nochmal in das Textfeld eingeben zu müssen (z.B. bei der Abfrage nach dem Betriebssystem für die Einrichtung des WLAN, sodass man einfach nur auf "macOS", "Android", etc. klicken muss)
10	no
11	Great work - you guys are amazing!
12	I would like to upload screenshots and error messages to the bot. Also for instructions the screenshots on the website are helpful, they would be nice too. And mostly but sometimes not the bot provides the source. It would always be helpful.
13	maybe be able to add pictures from the websites could be nice for Instructions, embedding to uni websites would fit good in my "search Flow", but that is not diretly chatbot related

Table 25 Responses to Question 12 – Suggestions and Issues - Part 1

ID	Response
14	I would love a more technical option of the bot/something that does not give the feel of having to have a converstation with a machine/something that is better suited to finding answers with as few keystrokes/clicks as possible (not having to write sentences or questions etc.)
15	Sources need to be reliable. Answer should be complete.
16	No nice idea.
17	I would prefer "I do not know" sooner in a conversation. I asked for example "I need a CRIS account" and received a wrong answer that would cause me a lot of questions to the wrong humans (plus a correct link to the CRIS intranet pages - but a) users are lazy and many will not click the additional link, b) the CRIS manual would be better to answer the question than the intranet :)). Only asking more specific "How does the CRIS of the university of Münster manage CRIS accounts?" I received "I do not know" (plus again the correct link to the CRIS intranet pages). Also more training / background data than only the CIT portal. And if using the CIT Portal as basic data, the answers of the chatbot should match the CIT pages I as an user would find on my own when skipping through the CIT portal. Otherwise the chatbot seems to have some secret knowledge the CIT is hiding. This could cause mistrust in the chatbot and / or the CIT
18	Ich habe den Chatbot nicht ausführlich genug getestet, er schien seine Aufgabe aber ausreichend gut zu erfüllen.

Table 26 Responses to Question 12 – Suggestions and Issues - Part 2

C.3 IT Support Staff Study:

Evaluation Question	Assigned Metric
The chatbot is able to provide complete and accurate responses to IT-related questions.	Accuracy
The chatbot was able to retain context and remember previous messages throughout the conversation.	Chat History
The chatbot understood and responded like a human would when answering questions.	Anthropomorphism
The chatbot's conversational style is similar to a real IT support person.	Personas
The chatbot is capable to resolve IT issues like the human support personnel.	Operational Efficiency
I believe that the chatbot can be part of the 1st-Level Support from now on.	Operational Efficiency
I believe that it could relieve or compensate for personnel shortages.	Operational Efficiency
I believe the chatbot will have a huge adoption of users.	Adoption
The chatbot's design and personality made it more engaging to interact with.	User Experience (UX) / Personas
The chatbot could be a beneficial way to increase the availability of IT Support.	Availability
The chatbot provides an easier way finding sources on the CIT website than searching by yourself.	Usability
Do you think the chatbot can help reduce the number of repeated requests of frequently asked questions?	Operational Efficiency
Do you think the chatbot is able to completely substitute the Support Hotline?	Operational Efficiency and Availability

Table 27 Mapping of IT Staff Evaluation Questions to Assessment Metrics

ID	Response
1	Überraschend gute Handlungsempfehlungen auch bei einem nicht alltäglichen Problem
2	You can ask it again and again, without it gets emotional
3	Pretty fast
4	schnelle und verschiedene Anwortmöglichkeiten
5	I like the detailed solutions and the quick way to get answers to frequently asked questions. Also the ability to get support at any time of the day.
6	Gut strukturierte (Schritt-für-Schritt) Lösungsvorschläge.
7	It includes contact information of the relevant IT support personnel.
8	The design is nice and with a bit of repeated asking it seems to sometimes give reasonably correct answers.

Table 28 IT Staff Feedback Question 2 – What They Liked About the Chatbot

ID	Response
1	Bei mehrfachen Fragen ist nicht intuitiv klar, wie man mehrere davon beantwortet. Oder ggf welche davon
2	If poorly programmed and you don't know the keywords, you're just standing in one place wasting your nerves and not getting the problem solved.
3	friendly
4	-
5	Manchmal etwas zu allgemein.
6	It very confidently gives incorrect information. For example, when asked about the installation of a niche program it simply assumed that the software follows the configure, make, make install process, which in reality it does not. I would prefer if the chatbot was more cautious about information it does not know reliably.
7	Er sucht immer eine Quelle, teilweise auch für Antworten für die es keine Quelle bedarf. Passiert dis, ist die Quelle manchmal random
8	All LLMs are prone to hallucinations. You often have to ask multiple times to get a relevant answer.

Table 29 IT Staff Feedback Question 3 – What They Disliked About the Chatbot

ID	Response
1	"Einfache" Fragen (z.B. Passwort vergessen) kann man sicherlich damit lösen. Das ist dann i.W. ein komfortabler Ersatz für FAQ-Lesen. Komplexere Anfragen benötigen einen 2nd-Level Support.
2	The most basic ones, for which there is already a clear and unambiguous manual.
3	Standard questions, after training maybe some more, hopefully Sciebo, too.
4	Wo finde ich was, wie funktioniert xx, etc.
5	Basic tasks such as answering questions like "Why is my OTP not working?" can be answered. Deeper things like sending out special otps and changing numbers should remain a human task.
6	I think the chatbot is a good alternative to problems that can be answered with the FAQ of the various services. Without always-up-to-date knowledge and/or direct access to systems like the identity management, it cannot answer correctly user- or system-specific questions (e.g., Why can I not access the HPC system? Is software XYZ installed on the cluster?)
7	Manche Nutzende wissen nicht was Ihr Problem ist und können die Probleme nicht korrekt bezeichnen, da wird der Bot sicherlich an seine Grenzen stoßen. Zudem gibt es so stark zeitbasierte Ereignisse, die unter den Kollegin*innen weitergeleitet werden, dass der Bot diese Anfragen zu diesen Themen nicht abfangen kann. Video-Idents oder generell die Prüfung der Identität.

Table 30 IT Staff Responses Question 4 – Types of Requests Suitable for Chatbot Support

ID	Response
1	Ist sicherlich eine Möglichkeit, außerhalb der Service-Zeiten einen 1st-Level-Support anzubieten. Ob das besser angenommen wird als der Hinweis auf die FAQ, bleibt abzuwarten.
2	Some customers are just more comfortable receiving information in quanta as it is given out by a chatbot and are not comfortable talking to a real person.
3	24/7 availability
4	Entlastung der Mitarbeiter bei "einfachen" Support Anfragen
5	Quicker response for customers and a decrease in repetitive questions.
6	I am doubtful of whether the chatbot will provide significant resource savings overall as the time savings due to successful interactions will have to compensate the additional time needed to support users that get bad advice from the bot. However, availability of some level of support outside of normal office hours is a plus that some customers might appreciate.
7	Ich denke, dass der Bot auf die Gewohnheiten und Bedürfnisse für Studierende angepasst ist. Diese haben häufig auch spät abends noch Fragen oder telefonieren nicht mehr so gerne. Gerade Anfragen zum Semesterstart können so besser abgefangen werden, hier werden immer wieder ähnliche Fragen gestellt zur Einrichtung des WLANs oder VPNs etc.
8	I expect redirection of human resources into unemployment, leading to overall lower customer satisfaction.

Table 31 IT Staff Responses to Question 5 – Expected Economical or Operative Effects

ID	Response
1	Die Datenbasis kann sicherlich noch erweitert werden.
2	The bot doesn't memorize answers, customers often tell you how to solve their problem themselves when they understand the nature of the problem during the conversation. Such answers should be memorized.
3	Sciebo training is essential for use in our queue.
4	Once I wasn't asked which operating system I was using and it just spit out the solution for Windows. However, I could not replicate this problem again.
5	The chatbot appears to just guess the sources of its information. I asked questions related to the HPC cluster and got somewhat sensible answers. However, I assume that the chatbot does not have access to the cluster documentation yet as it pointed to seemingly random other pages as its source. A bit more honesty in that regard would be nice ;-)
6	Mehr Auswahl in den Themen der IT

Table 32 IT Staff Responses to Question 6 - Suggestions for Improvement and Limitations

References

Adamopoulou, E., & Moussiades, L. (2020). Chatbots: History, technology, and applications. *Machine Learning with applications*, *2*, 100006.

Adnan, K., & Akbar, R. (2019). An analytical study of information extraction from unstructured and multidimensional big data. *Journal of Big Data*, *6*(1), 1–38.

Afzal, A., Kowsik, A., Fani, R., & Matthes, F. (2024). Towards optimizing and evaluating a retrieval augmented qa chatbot using llms with human in the loop. *arXiv preprint arXiv:2407.05925*.

AGI, M. (2024). *Clickup's transformation with maven agi: Elevating customer support and team efficiency* [Accessed: 2025-04-16]. https://www.mavenagi.com/resources/post/clickups-transformation-with-maven-agi-elevating-customer-support-and-team-efficiency

Akbar, N. A., Darmayanti, I., Fati, S. M., & Muneer, A. (2021). Deep learning of a pre-trained language model's joke classifier using gpt-2. *Journal of Hunan University Natural Sciences*, *48*(8).

Allen, D. W., Berg, C., Ilyushina, N., & Potts, J. (2023). Large language models reduce agency costs. *Available at SSRN 4437679*.

Anthropic. (2023, March). Introducing claude [Anthropic Blog, March 14]. https://www.anthropic.com/news/introducing-claude

Antico, C., Giordano, S., Koyuturk, C., & Ognibene, D. (2024). Unimib assistant: Designing a student-friendly rag-based chatbot for all their needs. *arXiv preprint arXiv:2411.19554*.

Arabi, A. F. M., Koyuturk, C., O'Mahony, M., Calati, R., & Ognibene, D. (2024). Habit coach: Customising rag-based chatbots to support behavior change. *arXiv preprint arXiv:2411.19229*.

Arslan, M., Ghanem, H., Munawar, S., & Cruz, C. (2024). A survey on rag with llms. *Procedia Computer Science*, *246*, 3781–3790.

Awais, M., Naseer, M., Khan, S., Anwer, R. M., Cholakkal, H., Shah, M., Yang, M.-H., & Khan, F. S. (2025). Foundation models defining a new era in vision: A survey and outlook. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Bansal, H., & Khan, R. (2018). A review paper on human computer interaction. *International Journal of Advanced Research in Computer Science and Software Engineering*, *8*(4), 53.

Bartl, A., & Spanakis, G. (2017). A retrieval-based dialogue system utilizing utterance and context embeddings. *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, 1120–1125.

Bhat, V., Cheerla, S. D., Mathew, J. R., Pathak, N., Liu, G., & Gao, J. (2024). Retrieval augmented generation (rag) based restaurant chatbot with ai testability. *2024 IEEE 10th International Conference on Big Data Computing Service and Machine Learning Applications (BigDataService)*, 1–10.

Caldarini, G., Jaf, S., & McGarry, K. (2022). A literature survey of recent advances in chatbots. *Information*, *13*(1), 41.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. *European conference on computer vision*, 213–229.

Carrick, D., & Kesteven, S. (2023, June 24). *This us lawyer used chatgpt to research a legal brief with embarrassing results. we could all learn from his error* [ABC News, with additional reporting from Reuters. Accessed: 2025-04-08]. https://www.abc.net.au/news/2023-06-24/us-lawyer-uses-chatgpt-to-research-case-with-embarrassing-result/102490068

Celikyilmaz, A., Clark, E., & Gao, J. (2020). Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*.

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

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Chroma. (2024). Chroma is the open-source ai application database. batteries included. [Accessed: 2025-04-18]. https://www.trychroma.com/

Colby, K. M. (1981). Parrying. *Behavioral and Brain Sciences*, *4*(4), 550–560.

Colby, K. M., Hilf, F. D., Weber, S., & Kraemer, H. C. (1972). Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes. *Artificial Intelligence*, *3*, 199–221.

Corea, F. (2019). Ai knowledge map: How to classify ai technologies. In F. Corea (Ed.), *An introduction to data* (pp. 25–29, Vol. 50). Springer International Publishing. https://doi.org/10.1007/978-3-030-04468-8_4

Cui, J., Ning, M., Li, Z., Chen, B., Yan, Y., Li, H., Ling, B., Tian, Y., & Yuan, L. (2023). Chatlaw: A multi-agent collaborative legal assistant with knowledge graph enhanced mixture-of-experts large language model. *arXiv preprint arXiv:2306.16092*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *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)*, 4171–4186.

Diao, S., Wang, P., Lin, Y., Pan, R., Liu, X., & Zhang, T. (2023). Active prompting with chain-of-thought for large language models. *arXiv:2302.12246*.

Dictionary.com. (2025). Chatbot definition [Accessed: 3 March 2025]. https://www.dictionary.com/browse/chatbot

Dong, Z., Tang, T., Li, L., & Zhao, W. X. (2023). A survey on long text modeling with transformers. *arXiv preprint arXiv:2302.14502*.

Essop, L., Singh, A., & Wing, J. (2023). Developing a comprehensive evaluation questionnaire for university faq administration chatbots. *2023 Conference on Information Communications Technology and Society (ICTAS)*, 1–7.

Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.-S., & Li, Q. (2024). A survey on rag meeting llms: Towards retrieval-augmented large language models. *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 6491–6501.

Flags, S. (2023). *Six flags to accelerate digital transformation in amusement park industry and launch a cutting edge virtual assistant powered by google cloud's generative ai* [Accessed: 2025-04-16]. https://investors.sixflags.com/news/press-releases/press-release-details/2023/news-08-29-2023-131412238/default.aspx

Floridi, L., & Chiriatti, M. (2020). Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, *30*, 681–694.

Fu, T., Gao, S., Zhao, X., Wen, J.-r., & Yan, R. (2022). Learning towards conversational ai: A survey. *AI Open*, *3*, 14–28.

Fujii, T., Shibata, K., Yamaguchi, A., Morishita, T., & Sogawa, Y. (2023). How do different tokenizers perform on downstream tasks in scriptio continua languages?: A case study in japanese. *arXiv preprint arXiv:2306.09572*.

Gamage, G., Mills, N., De Silva, D., Manic, M., Moraliyage, H., Jennings, A., & Alahakoon, D. (2024). Multi-agent rag chatbot architecture for decision support in net-zero emission energy systems. *2024 IEEE International Conference on Industrial Technology (ICIT)*, 1–6.

Gao, L., Ma, X., Lin, J., & Callan, J. (2023). Precise zero-shot dense retrieval without relevance labels. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1762–1777.

Ghojogh, B., & Ghodsi, A. (2020). Attention mechanism, transformers, bert, and gpt: Tutorial and survey.

Gu, Y., Tinn, R., Cheng, H., Lucas, M., Usuyama, N., Liu, X., Naumann, T., Gao, J., & Poon, H. (2021). Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, *3*(1), 1–23.

Hadi, M. U., Qureshi, R., Shah, A., Irfan, M., Zafar, A., Shaikh, M. B., Akhtar, N., Wu, J., Mirjalili, S., et al. (2023). Large language models: A comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea Preprints*, *1*, 1–26.

Howell, K., Christian, G., Fomitchov, P., Kehat, G., Marzulla, J., Rolston, L., Tredup, J., Zimmerman, I., Selfridge, E., & Bradley, J. (2023). The economic trade-offs of large language models: A case study. *arXiv preprint arXiv:2306.07402*.

Hsueh, P.-Y., Moore, J., & Renals, S. (2006). Automatic segmentation of multiparty dialogue. *11th Conference of the European Chapter of the Association for Computational Linguistics*, 273–280.

Hu, B., Lu, Z., Li, H., & Chen, Q. (2014). Convolutional neural network architectures for matching natural language sentences. *Advances in neural information processing systems*, *27*.

Huang, M., Zhu, X., & Gao, J. (2020). Challenges in building intelligent open-domain dialog systems. *ACM Transactions on Information Systems (TOIS)*, *38*(3), 1–32.

Hussain, S., Ameri Sianaki, O., & Ababneh, N. (2019). A survey on conversational agents/chatbots classification and design techniques. *Web, Artificial Intelligence and Network Applications: Proceedings of the Workshops of the 33rd International Conference on Advanced Information Networking and Applications (WAINA-2019) 33*, 946–956.

IBM. (2024). *Driving a reimagined customer experience with an ai-powered virtual assistant* [Accessed: 2025-04-16]. https://www.ibm.com/case-studies/camping-world

IONOS. (2024, July 4). *Python pickle: So serialisieren sie objekte in der programmiersprache* [Accessed: 2025-04-19]. https://www.ionos.de/digitalguide/websites/web-entwicklung/python-pickle/

Jia, J. (2009). Csiec: A computer assisted english learning chatbot based on textual knowledge and reasoning. *Knowledge-based systems*, *22*(4), 249–255.

Jurafsky, D. (2000). *Speech & language processing*. Pearson Education India.

Kaddour, J. (2023). The minipile challenge for data-efficient language models. *arXiv preprint arXiv:2304.08442*.

Kaiser, L., Bengio, S., Roy, A., Vaswani, A., Parmar, N., Uszkoreit, J., & Shazeer, N. (2018). Fast decoding in sequence models using discrete latent variables. *International Conference on Machine Learning*, 2390–2399.

Kang, L., Hu, B., Wu, X., Chen, Q., & He, Y. (2014). A short texts matching method using shallow features and deep features. *Natural Language Processing and Chinese Computing: Third CCF Conference, NLPCC 2014, Shenzhen, China, December 5-9, 2014. Proceedings 3*, 150–159.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Katz, D. M., Bommarito, M. J., Gao, S., & Arredondo, P. (2024). Gpt-4 passes the bar exam. *Philosophical Transactions of the Royal Society A*, *382*(2270), 20230254.

Khatri, C., Hedayatnia, B., Venkatesh, A., Nunn, J., Pan, Y., Liu, Q., Song, H., Gottardi, A., Kwatra, S., Pancholi, S., et al. (2018). Advancing the state of the art in open domain dialog systems through the alexa prize. *arXiv preprint arXiv:1812.10757*.

Labonne, M., & Moran, S. (2023). Spam-t5: Benchmarking large language models for few-shot email spam detection. *arXiv preprint arXiv:2304.01238*.

LangChain. (2024a). Build a retrieval augmented generation (rag) app [Copyright © 2024 LangChain, Inc.]. https://python.langchain.com/v0.2/docs/tutorials/rag/

LangChain. (2024b). Build an agent [Accessed: 2025-04-18]. https://python.langchain.com/docs/tutorials/agents/

LangChain. (2023a). History_aware_retriever.create_history_aware_retriever [Accessed: 2025-04-19]. https://api.python.langchain.com/en/latest/chains/langchain.chains.history_aware_retriever.create_history_aware_retriever.html

LangChain. (2024c). Langchain - introduction (v0.2) [Accessed: 2025-04-18]. https://python.langchain.com/v0.2/docs/introduction/

LangChain. (2024d). Langchain expression language (lcel) [Accessed: 2025-04-18]. https://python.langchain.com/v0.2/docs/concepts/#langchain-expression-language-lcel

LangChain. (2025a). *Select by maximal marginal relevance (mmr)* [Accessed: 2025-04-19]. https://python.langchain.com/v0.1/docs/modules/model_io/prompts/example_selectors/mmr/

LangChain. (2025b). *Similarity score threshold* [Accessed: 2025-04-19]. https://js.langchain.com/v0.1/docs/modules/data_connection/retrievers/similarity-score-threshold-retriever/

LangChain. (2023b). Source code for langchain_core.*vectorstores.base* [Accessed: 2025-04-19]. https://api.python.langchain.com/en/latest/_modules/langchain_core/vectorstores/base.html#VectorStore.similarity_search

Langchain. (2024a). How to use the multiqueryretriever [Accessed: 2025-04-13]. https://python.langchain.com/docs/how_to/MultiQueryRetriever/

Langchain. (2024b). Similarity metrics [Accessed: 2025-04-13]. https://python.langchain.com/docs/concepts/vectorstores/#similarity-metrics

Lee, J., Mansimov, E., & Cho, K. (2018). Deterministic non-autoregressive neural sequence modeling by iterative refinement. *arXiv preprint arXiv:1802.06901*.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, *33*, 9459–9474.

Liu, C.-W., Lowe, R., Serban, I. V., Noseworthy, M., Charlin, L., & Pineau, J. (2016). How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

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

Lu, Z., & Li, H. (2013). A deep architecture for matching short texts. *Advances in neural information processing systems*, *26*.

Luitse, D., & Denkena, W. (2021). The great transformer: Examining the role of large language models in the political economy of ai. *Big Data & Society*, *8*(2), 20539517211047734.

Luo, R., Zhao, Z., Yang, M., Dong, J., Li, D., Lu, P., Wang, T., Hu, L., Qiu, M., & Wei, Z. (2023). Valley: Video assistant with large language model enhanced ability. *arXiv preprint arXiv:2306.07207*.

Marietto, M. d. G. B., de Aguiar, R. V., Barbosa, G. d. O., Botelho, W. T., Pimentel, E., França, R. d. S., & da Silva, V. L. (2013). Artificial intelligence markup language: A brief tutorial. *arXiv preprint arXiv:1307.3091*.

Microsoft. (2024). *Telstra dials in elevated customer service with azure openai service* [Accessed: 2025-04-16]. https://www.microsoft.com/en/customers/story/1740058425924206437-telstra-telecommunications-azure-openai-service

moinAI. (2025a). *Ai chatbot deployed within 14 days* [Accessed: 2025-04-16]. https://www.moin.ai/en/case-studies/case-study-teag-chatbot

moinAI. (2023). Ai chatbot for b2b and b2c marketing. https://www.moin.ai/en/case-studies/geberit-chatbot

moinAI. (2025b). *Ai chatbot for fast and excellent first-level support* [Accessed: 2025-04-16]. https://www.moin.ai/en/case-studies/ai-chatbot-for-fast-und-excellent-first-level-support

moinAI. (2025c). *Ai chatbot for more efficiency in customer service and marketing* [Accessed: 2025-04-16]. https://www.moin.ai/en/case-studies/ai-chatbot-efficiency-customer-service-marketing

moinAI. (2025d). *How faz uses an ai chatbot for service requests* [Accessed: 2025-04-16]. https://www.moin.ai/en/case-studies/chatbot-faz-publisher

Mullick, S. S., Bhambhani, M., Sinha, S., Mathur, A., Gupta, S., & Shah, J. (2023). Content moderation for evolving policies using binary question answering. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, 561–573.

Murel, J., & Noble, J. (2024, October 1). *What is an encoder-decoder model?* [Accessed: 2025-04-05]. https://www.ibm.com/think/topics/encoder-decoder-model

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., & Mian, A. (2023). A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*.

Olawore, K., McTear, M., & Bi, Y. (2023). Development and evaluation of a university chatbot using deep learning: A rag-based approach. *Chatbots and Human-Centered AI*, 96.

Olujimi, P. A., & Ade-Ibijola, A. (2023). Nlp techniques for automating responses to customer queries: A systematic review. *Discover Artificial Intelligence*, *3*(1), 20.

OpenAI. (2025). Gpt-4 is openai's most advanced system, producing safer and more useful responses [Accessed: 2025-04-18]. https://openai.com/index/gpt-4/

OpenAI. (2024a). Gpt-4o [Accessed: 2025-04-13]. https://platform.openai.com/docs/models/gpt-4o

OpenAI. (2024b, January). New embedding models and api updates [Accessed: 2025-04-18]. https://openai.com/index/new-embedding-models-and-api-updates/?utm_source=chatgpt.com

OpenAI. (2024c). Openai platform - embeddings [Accessed: 2025-04-13]. https://platform.openai.com/docs/guides/embeddings

OpenAI. (2024d). *Tokenizer* [Accessed: 2025-04-05]. https://platform.openai.com/tokenizer

Pichai, S., & Hassabis, D. (2023). Introducing gemini: Our largest and most capable ai model [Google Blog, December 6]. https://blog.google/technology/ai/google-gemini-ai/#sundar-note

Quidwai, M. A., & Lagana, A. (2024). A rag chatbot for precision medicine of multiple myeloma. *medRxiv*, 2024–03.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023). Robust speech recognition via large-scale weak supervision. *International conference on machine learning*, 28492–28518.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, *21*(140), 1–67.

Ramesh, K., Ravishankaran, S., Joshi, A., & Chandrasekaran, K. (2017). A survey of design techniques for conversational agents. *International conference on information, communication and computing technology*, 336–350.

Ramírez, J. G. C. (2024). Natural language processing advancements: Breaking barriers in human-computer interaction. *Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023*, *3*(1), 31–39.

Richardson, L. (2004–2025). Beautiful soup documentation [Accessed: 2025-04-18]. https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Rosset, C., Chung, H.-L., Qin, G., Chau, E. C., Feng, Z., Awadallah, A., Neville, J., & Rao, N. (2024). Researchy questions: A dataset of multi-perspective, decompositional questions for llm web agents. *arXiv preprint arXiv:2402.17896*.

Ruder, S., Peters, M. E., Swayamdipta, S., & Wolf, T. (2019). Transfer learning in natural language processing. *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Tutorials*, 15–18.

Sabry, A. (2023). *A perfect guide to understand encoder decoders in depth with visuals* [Accessed: 2025-04-05]. https://medium.com/@ahmadsabry678/a-perfect-guide-to-understand-encoder-decoders-in-depth-with-visuals-30805c23659b

Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2017). Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*.

Schwartz, R., Dodge, J., Smith, N. A., & Etzioni, O. (2020). Green ai. *Communications of the ACM*, *63*(12), 54–63.

Sennrich, R., Haddow, B., & Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Shang, L., Lu, Z., & Li, H. (2015). Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Shaw, P., Uszkoreit, J., & Vaswani, A. (2018). Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Shieber, S. M. (1994). Lessons from a restricted turing test. *arXiv preprint cmp-lg/9404002*.

Shum, H.-Y., He, X.-d., & Li, D. (2018). From eliza to xiaoice: Challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, *19*, 10–26.

Shuster, K., Poff, S., Chen, M., Kiela, D., & Weston, J. (2021). Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*.

Singh, S. U., & Namin, A. S. (2025). A survey on chatbots and large language models: Testing and evaluation techniques. *Natural Language Processing Journal*, 100128.

Sinha, G., Shahi, R., & Shankar, M. (2010). Human computer interaction. *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, 1–4.

Snowflake Inc. (2025a). Deploy streamlit using docker [Accessed: 2025-04-18]. https://docs.streamlit.io/deploy/tutorials/docker

Snowflake Inc. (2025b). Deploy streamlit using kubernetes [Accessed: 2025-04-18]. https://docs.streamlit.io/deploy/tutorials/kubernetes

Snowflake Inc. (2025c). Embed your app [Accessed: 2025-04-19]. https://docs.streamlit.io/deploy/streamlit-community-cloud/share-your-app/embed-your-app

Snowflake Inc. (2025d). St.cache_resource [Accessed: 2025-04-18]. https://docs.streamlit.io/develop/api-reference/caching-and-state/st.cache_resource

Snowflake Inc. (2025e). Streamlit - a faster way to build and share data apps [Accessed: 2025-04-18].

Snowflake Inc. (2025f). Streamlit documentation [Accessed: 2025-04-18]. https://docs.streamlit.io/

Spherical Insights & Consulting. (2024). Global Chatbot Market Size To Exceed USD 42.83 Billion By 2033 | CAGR of 23.03% [Accessed: 2024-04-22]. https://finance.yahoo.com/news/global-chatbot-market-size-exceed-080000758.html

Strubell, E., Ganesh, A., & McCallum, A. (2020). Energy and policy considerations for modern deep learning research. *Proceedings of the AAAI conference on artificial intelligence*, *34*(09), 13693–13696.

Su, D., Xu, Y., Winata, G. I., Xu, P., Kim, H., Liu, Z., & Fung, P. (2019). Generalizing question answering system with pre-trained language model fine-tuning. *Proceedings of the 2nd workshop on machine reading for question answering*, 203–211.

Suhaili, S. M., Salim, N., & Jambli, M. N. (2021). Service chatbots: A systematic review. *Expert Systems with Applications*, *184*, 115461.

Susanto, E., & Khaq, Z. D. (2024). Enhancing customer service efficiency in start-ups with ai: A focus on personalization and cost reduction. *Journal of Management and Informatics*, *3*(2), 267–281.

Tamilmani, K., Rana, N. P., Wamba, S. F., & Dwivedi, R. (2021). The extended unified theory of acceptance and use of technology (utaut2): A systematic literature review and theory evaluation. *International Journal of Information Management*, *57*, 102269. https://doi.org/10.1016/j.ijinfomgt.2020.102269

Terveer, I. (2019). *Mathematik für wirtschaftswissenschaften*. utb GmbH.

The Linux Foundation. (2025a). Kubernetes [Accessed: 2025-04-19]. https://kubernetes.io/docs/tutorials/kubernetes-basics/

The Linux Foundation. (2025b). Kubernetes [Accessed: 2025-04-19]. https://kubernetes.io/

The Linux Foundation. (2025c). Kubernetes [Accessed: 2025-04-19]. https://kubernetes.io/docs/concepts/overview/

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Turing, A. M. (2009). *Computing machinery and intelligence*. Springer.

University of Münster. (2025a). Kubernetes documentation [Accessed: 2025-04-19]. https://cloud.uni-muenster.de/docs/kubernetes/

University of Münster. (2025b). Kubernetes documentation [Accessed: 2025-04-19]. https://cloud.uni-muenster.de/docs/kubernetes/centralcomponents/

University of Münster. (2025c). Kubernetes documentation [Accessed: 2025-04-19]. https://cloud.uni-muenster.de/docs/kubernetes/environmentsandlocations/

Vakayil, S., Juliet, D. S., Vakayil, S., et al. (2024). Rag-based llm chatbot using llama-2. *2024 7th International Conference on Devices, Circuits and Systems (ICDCS)*, 1–5.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Volum, R., Rao, S., Xu, M., DesGarennes, G., Brockett, C., Van Durme, B., Deng, O., Malhotra, A., & Dolan, W. B. (2022). Craft an iron sword: Dynamically generating interactive game characters by prompting large language models tuned on code. *Proceedings of the 3rd Wordplay: When Language Meets Games Workshop (Wordplay 2022)*, 25–43.

Wallace, R. S. (2009). *The anatomy of alice*. Springer.

Wang, L., Yang, N., & Wei, F. (2023). Query2doc: Query expansion with large language models. *arXiv preprint arXiv:2303.07678*.

Wang, L., Lyu, C., Ji, T., Zhang, Z., Yu, D., Shi, S., & Tu, Z. (2023). Document-level machine translation with large language models. *arXiv:2304.02210*.

Wang, M., Lu, Z., Li, H., & Liu, Q. (2015). Syntax-based deep matching of short texts. *arXiv preprint arXiv:1503.02427*.

Wang, X., Wang, Z., Gao, X., Zhang, F., Wu, Y., Xu, Z., Shi, T., Wang, Z., Li, S., Qian, Q., et al. (2024). Searching for best practices in retrieval-augmented generation. *arXiv preprint arXiv:2407.01219*.

Wang, Z., Li, S., Chen, G., & Lin, Z. (2017). Deep and shallow features learning for short texts matching. *2017 International Conference on Progress in Informatics and Computing (PIC)*, 51–55.

Wang, Z., Wang, Z., Long, Y., Wang, J., Xu, Z., & Wang, B. (2019). Enhancing generative conversational service agents with dialog history and external knowledge. *Computer Speech & Language*, *54*, 71–85.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. (2022). Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, *35*, 24824–24837.

Wei, J., Kim, S., Jung, H., & Kim, Y.-H. (2024). Leveraging large language models to power chatbots for collecting user self-reported data. *Proceedings of the ACM on Human-Computer Interaction*, *8*(CSCW1), 1–35.

Wei, X., Cui, X., Cheng, N., Wang, X., Zhang, X., Huang, S., Xie, P., Xu, J., Chen, Y., Zhang, M., et al. (2023). Chatie: Zero-shot information extraction via chatting with chatgpt. *arXiv preprint arXiv:2302.10205*.

Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *9*(1), 36–45.

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.

Wieringa, R. (2014). *Design science methodology for information systems and software engineering*. Springer.

Wu, S., Xiong, Y., Cui, Y., Wu, H., Chen, C., Yuan, Y., Huang, L., Liu, X., Kuo, T.-W., Guan, N., et al. (2024). Retrieval-augmented generation for natural language processing: A survey. *arXiv preprint arXiv:2407.13193*.

Wu, X.-K., Chen, M., Li, W., Wang, R., Lu, L., Liu, J., Hwang, K., Hao, Y., Pan, Y., Meng, Q., et al. (2025). Llm fine-tuning: Concepts, opportunities, and challenges. *Big Data and Cognitive Computing*, *9*(4), 87.

Wu, Y., Wu, W., Xing, C., Zhou, M., & Li, Z. (2016). Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. *arXiv preprint arXiv:1612.01627*.

Yang, L., Qiu, M., Qu, C., Guo, J., Zhang, Y., Croft, W. B., Huang, J., & Chen, H. (2018). Response ranking with deep matching networks and external knowledge in information-seeking conversation systems. *The 41st international acm sigir conference on research & development in information retrieval*, 245–254.

Yin, W., Kann, K., Yu, M., & Schütze, H. (2017). Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q., & Liu, Z. (2024). Evaluation of retrieval-augmented generation: A survey. *CCF Conference on Big Data*, 102–120.

Zendesk. (2024). *Motel sees self-service climb by 3x and faster ticket handling with zendesk ai* [Accessed: 2025-04-16]. https://www.zendesk.com/customer/motel-rocks/

Zeng, W., Ren, X., Su, T., Wang, H., Liao, Y., Wang, Z., Jiang, X., Yang, Z., Wang, K., Zhang, X., et al. (2021). Pangu-alpha: Large-scale autoregressive pretrained chinese language models with auto-parallel computation. *arXiv preprint arXiv:2104.12369*.

Zhang, T., Ladhak, F., Durmus, E., Liang, P., McKeown, K., & Hashimoto, T. B. (2024). Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, *12*, 39–57.

Zhang, W., Deng, Y., Liu, B., Pan, S. J., & Bing, L. (2023). Sentiment analysis in the era of large language models: A reality check. *arXiv preprint arXiv:2305.15005*.

Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., Yang, L., Zhang, W., Jiang, J., & Cui, B. (2024). Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*, *1*(2).

Zheng, H. S., Mishra, S., Chen, X., Cheng, H.-T., Chi, E. H., Le, Q. V., & Zhou, D. (2023). Take a step back: Evoking reasoning via abstraction in large language models. *arXiv preprint arXiv:2310.06117*.

Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., Zhou, D., & Hou, L. (2023). Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Zhou, X., Li, L., Dong, D., Liu, Y., Chen, Y., Zhao, W. X., Yu, D., & Wu, H. (2018). Multi-turn response selection for chatbots with deep attention matching network. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1118–1127.

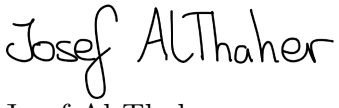
Zhu, D., Chen, J., Shen, X., Li, X., & Elhoseiny, M. (2023). Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*.

Zhuo, T. Y., Li, Z., Huang, Y., Shiri, F., Wang, W., Haffari, G., & Li, Y.-F. (2023). On robustness of prompt-based semantic parsing with large pre-trained language model: An empirical study on codex. *arXiv preprint arXiv:2301.12868*.

Declaration of Authorship

I hereby declare that, to the best of my knowledge and belief, this Master Thesis titled "Exploring Retrieval-Augmented Generation for Large Language Models: Enhancing University IT Support with a RAG-based Chatbot" is my own work. I confirm that each significant contribution to and quotation in this thesis that originates from the work or works of others is indicated by proper use of citation and references.

Münster, 22nd April 2025



Josef Al Thaher

Consent Form

for the use of plagiarism detection software to check my thesis

Last name: Al Thaher **First name**: Josef
Student number: MS:430410, UT:3268225 **Course of study**: Information Systems
Address: Weseler Straße 569, 48163 Münster
Title of the thesis: "Exploring Retrieval-Augmented Generation for Large Language Models: Enhancing University IT Support with a RAG-based Chatbot"

What is plagiarism? Plagiarism is defined as submitting someone else's work or ideas as your own without a complete indication of the source. It is hereby irrelevant whether the work of others is copied word by word without acknowledgment of the source, text structures (e.g. line of argumentation or outline) are borrowed or texts are translated from a foreign language.

Use of plagiarism detection software The examination office uses plagiarism software to check each submitted bachelor and master thesis for plagiarism. For that purpose the thesis is electronically forwarded to a software service provider where the software checks for potential matches between the submitted work and work from other sources. For future comparisons with other theses, your thesis will be permanently stored in a database. Only the School of Business and Economics of the University of Münster is allowed to access your stored thesis. The student agrees that his or her thesis may be stored and reproduced only for the purpose of plagiarism assessment. The first examiner of the thesis will be advised on the outcome of the plagiarism assessment.

Sanctions Each case of plagiarism constitutes an attempt to deceive in terms of the examination regulations and will lead to the thesis being graded as "failed". This will be communicated to the examination office where your case will be documented. In the event of a serious case of deception the examinee can be generally excluded from any further examination. This can lead to the exmatriculation of the student. Even after completion of the examination procedure and graduation from university, plagiarism can result in a withdrawal of the awarded academic degree.

I confirm that I have read and understood the information in this document. I agree to the outlined procedure for plagiarism assessment and potential sanctioning.

Münster, 22nd April 2025

Josef Al Thaher