Politecnico di Torino

Politecnico di Torino

Msc in Data Science and Engineering

Retrieval-Augmented Generation for Technical Documentation: A Domain-Specific Chatbot for Firmware Manuals

Candidate: Teresa Argnani

Supervisor: Prof. Elena Maria Baralis

December 2025

Contents

Introduction

This thesis was developed in collaboration with Brain Technologies, an engineering consulting company specializing in software engineering, firmware development, and control systems. Within this professional context, one recurring challenge emerged: firmware developers frequently need to consult extensive technical documentation, such as reference manuals, datasheets, and application notes, to correctly configure microcontrollers and ensure proper hardware-software integration. These documents are often long and complex, with information distributed across multiple sections, tables, and diagrams, or even across different manuals that may use inconsistent notation. As a result, finding a specific configuration detail or parameter can be a slow and error-prone process, consuming time that could otherwise be dedicated to actual development.

To address this difficulty, this work investigates the use of modern language technologies to help developers navigate such documentation more efficiently. In recent years, Large Language Models (LLMs) have gained remarkable attention for their ability to understand and generate natural language. Their rapid development and increasing accessibility have made them widely available tools for diverse applications, from creative writing to software assistance. However, despite their versatility, general-purpose LLMs often struggle with specialized or highly technical content, where precision, contextual understanding, and terminology consistency are crucial.

To address this limitation, the Retrieval-Augmented Generation (RAG) paradigm has emerged as a promising approach. By combining the generative capabilities of LLMs with the retrieval of relevant information from external knowledge sources, RAG systems are able to produce more accurate and contextually grounded responses. This strategy is particularly suitable for interacting with technical material, where factual correctness and traceability are essential.

The objective of this thesis is to explore the use of Retrieval-Augmented Generation

for developing a domain-specific chatbot designed to assist firmware engineers in consulting microcontroller manuals. The proposed system retrieves relevant sections from device documentation and generates concise, well-referenced answers to technical queries. Its development required addressing several key aspects of the problem. A significant part of the work focused on document preprocessing and structuring, as microcontroller manuals contain valuable information not only in text but also in tables and figures. Preserving this heterogeneous data required a robust and adaptable preprocessing pipeline capable of handling diverse document layouts. Subsequent steps involved the creation of semantic embeddings and a vector database for efficient information retrieval, the design of targeted prompts to guide the model's reasoning, and the evaluation of the resulting chatbot through case-based testing that measured its correctness, completeness, and usability. These experiments provided insights into the effectiveness of the RAG approach in managing technical documentation and highlighted practical challenges and directions for future improvement.

The thesis concludes with a discussion of the main findings and considerations for industrial application, including the challenges related to data preprocessing complexity, model accuracy, and computational cost.

Chapter 1

Background and Theoretical Foundations

This chapter provides the theoretical background necessary to understand the concepts and technologies behind the system developed in this thesis. It begins with an overview of chatbots, tracing their evolution from early rule-based programs to modern conversational systems powered by Large Language Models (LLMs). The discussion then introduces the main principles of LLMs, including their architecture, training process, and current limitations. Building on these foundations, the chapter presents the concept of Retrieval-Augmented Generation (RAG), a framework designed to improve factual accuracy and adaptability in domain-specific contexts. Finally, it explores the role of prompt engineering in guiding model behavior and ensuring that generated responses remain consistent with the source documentation. Together, these sections establish the conceptual basis for the development and evaluation presented in the following chapters.

1.1 Chatbots

Chatbots represent one of the most visible applications of artificial intelligence in everyday life. This section introduces their fundamental principles and traces their evolution from early rule-based systems to modern language-model-based assistants. It concludes with an overview of the main application domains where chatbots are employed today, establishing the background necessary to understand their role within the context of this thesis.

1.1.1 Chatbots Overview

According to the Oxford English Dictionary, a chatbot is defined as "A computer program designed to simulate conversation with a human user, usually over the internet" [1]. Over the years, various terms have been used interchangeably with "chatbot", including machine conversation system, virtual agent, dialogue system, and chatterbot [2]. Despite these variations in terminology, the underlying principle remains the same: enabling machines to interact with users in a way that resembles natural communication.

The earliest chatbots were rule-based, relying on predefined scripts, question-answer pairs, and pattern-matching techniques. These systems identified certain keywords in the user's input and returned predefined responses, thus creating the illusion of conversation. However, such systems were highly constrained and often produced repetitive or irrelevant replies when user inputs deviated from expected patterns [3].

A significant step forward was the introduction of intent-based chatbots, which leverage Natural Language Understanding (NLU) techniques to capture the underlying intent of a user's query rather than reacting only to surface-level keywords. This shift enabled more robust and flexible interactions, as responses could be dynamically selected based on the recognized intent [4]. For example, an intent-based system could interpret the inputs "What's the weather like tomorrow?" and "Will it rain tomorrow?" as expressing the same request for weather information and provide an appropriate answer.

Meanwhile, the exponential growth of digital data and advances in machine learning paved the way for more sophisticated conversational systems. The emergence of Large Language Models (LLMs) has been especially transformative. Trained on vast corpora of text, these models demonstrate a remarkable ability to understand linguistic nuances, maintain conversational context, and generate coherent, contextually appropriate responses [5]. LLM-based chatbots represent the most advanced stage of conversational AI to date, although challenges such as hallucinations, outdated knowledge, and computational cost remain open issues [6].

1.1.2 Evolution of Chatbots

The history of chatbots reflects the broader evolution of artificial intelligence and computational linguistics. Early conversational systems relied entirely on handcrafted rules, while modern chatbots employ data-driven and generative models. This progression has been shaped by advances in natural language processing, computing power, and

human-computer interaction.

The conceptual foundation for conversational agents is often traced to Alan Turing's 1950 paper *Computing Machinery and Intelligence* [7], which introduced the idea later known as the "Turing Test". Although not a chatbot itself, this thought experiment provided an early framework for evaluating machine intelligence in text-based communication.

A major milestone occurred in 1966 with the development of ELIZA by Joseph Weizenbaum [8]. ELIZA simulated a psychotherapist using simple pattern-matching rules and demonstrated how scripted responses could create the illusion of understanding. Soon after, PARRY (1972) [9] extended this idea by incorporating a rudimentary internal state to emulate a patient with schizophrenia, representing an early attempt to model psychological consistency.

The following decades saw experimentation with conversational systems in both research and entertainment contexts. Systems such as Jabberwacky (1988), TinyMUD (1991), and Dr. Sbaitso (1992) explored interactive dialogue in playful or multimedia environments. Although limited by hardware constraints and simple learning mechanisms, they broadened the scope of chatbot applications [10].

A notable advance came with ALICE (1995) and the introduction of AIML (Artificial Intelligence Markup Language) [11]. ALICE demonstrated the scalability of rule-based systems, storing tens of thousands of conversational templates and winning multiple Loebner Prizes [12]. However, it still lacked genuine language understanding, relying strictly on symbolic pattern matching.

The early 2000s saw the deployment of chatbots to mainstream platforms [13]. SmarterChild (2001) integrated with messaging services and provided fast, task-oriented responses, foreshadowing modern personal assistants. In 2008, Cleverbot popularized learning from user interactions, although it struggled with long-term coherence.

The 2010s marked the transition from scripted dialogue to large-scale statistical and neural approaches. Voice assistants such as Apple's Siri (2010), IBM's Watson (2011) [14], Google Now, Cortana, and Amazon's Alexa (2014) combined speech recognition, natural language understanding, and internet connectivity to provide contextual, multimodal interaction [15].

A fundamental shift occurred with the emergence of LLMs. OpenAI's GPT-3 (2020) [16] demonstrated that transformer-based architectures could generate coherent, context-

Year	System	Main contribution	Main limitation
1950	Turing Test	Conceptual framework for evaluating machine intelligence	Purely theoretical; no implementation
1966	ELIZA	First rule-based conversational program using pattern matching	No semantic understanding; scripted responses
1972	PARRY	Introduced persona modeling and internal state	Still rule-based; limited reasoning
1988	Jabberwacky	Early learning from user interactions; playful dialogue	No long-term coherence; hardware limitations
1995	ALICE	Introduced AIML; scalable template-based system	No true language understanding; symbolic matching
2001	SmarterChild	First mass-market chatbot integrated into IM platforms	Narrow domains; scripted logic
2008	Cleverbot	Web-scale conversational learning from users	Inconsistent replies; no context retention
2010-2014	Siri, Watson, Alexa	Voice-based multimodal assistants combining speech recognition, NLU, online services	Intent-based limits; shallow reasoning
2020+	GPT-3, ChatGPT, Gemini, LLaMA	Generative conversational AI with broad generalization and contextual reasoning	Hallucinations; high computational cost; knowledge staleness

Table 1.1: Key milestones in the evolution of chatbots

aware text without relying on predefined rules or handcrafted intent structures. This marked a decisive departure from earlier approaches: instead of mapping queries to fixed templates, LLMs learned broad linguistic patterns directly from large corpora, enabling open-ended and flexible dialogue. The release of ChatGPT (2022) [17] introduced conversational models capable of maintaining context, following instructions, and reasoning across diverse topics. Subsequent models, including Google's Gemini [18] and Meta's LLaMA [19], further expanded their abilities in multilingual processing, extended

context windows, and more sophisticated reasoning. These developments represent the transition from symbolic or task-specific systems to generative conversational AI and provide the basis for the techniques explored later in this thesis.

To summarize the main milestones, Table 1.1 highlights representative systems, the innovations they introduced, and their main limitations.

1.1.3 Chatbot Applications

Chatbots have evolved from experimental systems into widely adopted tools across many domains. Their ability to understand user intent, manage context, and generate natural language responses has made them valuable in settings ranging from education to business, healthcare, and everyday digital interaction. Although the specific functions differ across these areas, chatbot applications share common requirements such as reliability, contextual understanding, and the ability to process domain-specific information [10].

In education and research, chatbots can support learners through interactive exercises, personalized tutoring, and access to on-demand explanations. Language-learning systems can provide instant feedback and adapt to a student's proficiency level, while accessibility features such as text-to-speech and speech-to-text assist users with disabilities [20]. In academic research, conversational agents are increasingly used to summarize scientific papers, identify relevant literature, and automate repetitive information retrieval tasks.

Chatbots are also used as tools for creativity and idea generation. Their capacity to suggest alternative perspectives, explore conceptual connections, and synthesize diverse information makes them useful for brainstorming and early-stage design activities. In professional environments, conversational systems assist in project planning, content drafting, and refining initial concepts.

In software engineering and technical domains, chatbots powered by LLMs support programmers in tasks such as code generation, debugging, documentation, and testing. They can detect syntactic or logical errors, propose optimized algorithms, or explain unfamiliar pieces of code, thereby reducing development time and improving productivity [21]. These capabilities illustrate how conversational interfaces can help developers access complex technical information more efficiently, an aspect closely related to the focus of this thesis.

In business and public administration, chatbots enhance communication and customer engagement [22]. They handle frequent queries, guide users through product selection, provide technical support, and escalate complex issues to human operators when needed. Their continuous availability reduces response times and operational costs. Within public institutions, chatbots help citizens navigate administrative procedures, complete digital forms, or access online services.

In healthcare, conversational agents support tasks such as appointment scheduling, patient triage, and health education. Recent studies indicate that chatbots can provide preliminary medical guidance or answer clinical questions with reasonable accuracy, although human oversight remains essential for safety [23]. They are also employed to promote healthy behaviors, offer reminders, and deliver basic emotional support, particularly when integrated into telemedicine platforms.

Finally, in everyday life, chatbots are embedded in personal assistants, smart home devices, and productivity tools. Users rely on them to schedule activities, control connected devices, retrieve information, or compose messages through natural language commands. This widespread integration illustrates how conversational AI has become a common interface for managing routine digital interactions.

Overall, chatbot applications demonstrate the transformative potential of natural language interfaces in mediating human-computer interaction. By enabling more intuitive communication and reducing technical barriers, chatbots are reshaping how people access information and perform tasks. Despite their benefits, challenges related to contextual understanding, privacy, reliability, and ethical use persist [24], which motivates the use of techniques that improve robustness and factual accuracy, such as the retrieval-based approaches discussed in the following sections.

1.2 Large Language Models

Large Language Models (LLMs) have emerged as a central paradigm in Natural Language Processing (NLP). These models can generate contextually appropriate text, answer questions, and perform a wide range of language-related tasks. Their capabilities are made possible by advances in neural network architectures, efficient language representations, and the availability of large-scale training data. Within conversational systems, LLMs provide the contextual and linguistic understanding that enables modern chatbots to produce adaptive and human-like responses.

This section introduces the core concepts needed to understand the role of LLMs in conversational AI. It first describes how language is represented numerically inside these models, then examines the architectural principles that allow LLMs to capture complex dependencies in text. The discussion continues with an overview of the training strategies used to develop and adapt these models and concludes with their main applications and limitations. Together, these elements provide the conceptual foundations for the retrieval-based approach presented later in this thesis.

1.2.1 Word Embeddings and Representation Learning

A fundamental step in natural language processing is the transformation of words into numerical representations that can be processed by machine learning models. Early approaches such as one-hot encoding assigned each word a high-dimensional binary vector with a single active entry. Although straightforward, this representation treated every word as unrelated to all others, making it unable to capture semantic or syntactic similarities.

Word embeddings were introduced to address this limitation by mapping words into dense, low-dimensional vectors in which semantically related words lie closer together in the vector space. Methods such as Word2Vec [25] and GloVe [26] learned these vectors from large text corpora by analyzing word co-occurrence patterns. These embeddings also capture semantic structure in a geometric way: linguistic relationships such as gender, verb tense, or intensity often correspond to consistent directions or offsets in the vector space, revealing how meaningful patterns can emerge from statistical learning rather than explicit rules.

However, fixed embeddings cannot represent words that have different meanings in different contexts. To overcome this issue, models such as ELMo [27] and BERT [28] introduced contextual embeddings, where the vector associated with a word depends on the sentence in which it appears. This shift greatly improved the ability of NLP systems to capture polysemy, grammatical structure, and long-range dependencies.

Modern LLMs build on these advances by learning embedding representations jointly with the rest of the network during training. Instead of treating whole words as atomic units, they typically rely on subword tokenization methods such as Byte-Pair Encoding (BPE) or SentencePiece [29], which allow the model to represent rare words, morphological variations, and out-of-vocabulary terms efficiently. The resulting embedding layer

forms the foundation upon which deeper neural components model increasingly abstract linguistic structures.

1.2.2 Transformer Architecture

A major breakthrough in modern language modeling came with the introduction of the Transformer architecture [30]. Transformers rely entirely on attention mechanisms rather than sequential processing, allowing them to model dependencies between tokens regardless of their distance in the input sequence. This design enables efficient parallelization during training and improves the ability to capture long-range relationships within text.

At the core of the Transformer is the self-attention mechanism, which determines how much each token should attend to every other token in the sequence. Each token is projected into three vectors: a query (Q), a key (K), and a value (V). Attention scores are computed by comparing queries with keys, and these scores determine how the corresponding value vectors contribute to the updated token representation. The attention mechanism compares each query with all keys, normalizes the resulting scores through a softmax function, and uses them to compute a weighted sum of the value vectors:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

where d_k is the dimensionality of the key vectors. These attention scores act as relevance weights, allowing each token to form a context-aware representation that selectively integrates information from the entire sequence.

Transformers employ multiple attention heads operating in parallel, enabling the model to capture different types of linguistic relationships simultaneously. Additional architectural components further support stable and effective learning: residual connections facilitate information flow across layers, layer normalization improves training dynamics, and position-wise feed-forward networks provide non-linear transformations for each token.

Because self-attention alone does not encode sequence order, positional encodings are added to give the model information about the relative or absolute position of tokens. These encodings allow the Transformer to incorporate ordering information while maintaining full parallelization.

The original Transformer architecture follows an encoder-decoder structure. The

encoder reads the input sequence and produces contextual representations for each token. The decoder then generates the output one token at a time, attending both to previously generated tokens (through self-attention) and to the encoder representations (through cross-attention). This framework is particularly effective for sequence-to-sequence tasks such as machine translation or abstractive summarization. Many modern LLMs, such as GPT and its successors, instead adopt a decoder-only architecture optimized for autoregressive text generation, whereas models such as T5 [31] and BART [32] retain the full encoder-decoder design for tasks that require processing an input sequence and generating a separate output.

1.2.3 Training Process of LLMs

LLMs are trained using self-supervised learning, a paradigm in which the model learns directly from raw text without requiring manually annotated labels. In this setting, the training objective is derived from the text itself. Two main formulations are commonly used. In autoregressive language modeling, the model predicts the next token given all previous ones, as in the GPT family of models [33]. In masked language modeling, some input tokens are replaced by a special mask symbol, and the model learns to recover them from the surrounding context, as in BERT [28]. These objectives allow the model to acquire broad linguistic and world knowledge from large corpora in a scalable manner.

After this pre-training stage, models can be adapted to specific applications through fine-tuning. In supervised fine-tuning, the model is trained on labeled datasets so that it learns to perform a particular task, such as sentiment analysis, classification, or dialogue management. More recently, instruction tuning has been introduced, where models are fine-tuned on collections of task descriptions paired with expected responses to improve their ability to follow user instructions [34]. A further refinement is Reinforcement Learning from Human Feedback (RLHF) [35], in which human evaluators compare model outputs and these preferences are used to optimize the model toward more helpful and aligned behavior.

Although effective, full fine-tuning is computationally demanding and often impractical for large models. To reduce the cost, parameter-efficient fine-tuning techniques such as adapters or LoRA (Low-Rank Adaptation) [36] modify only a small subset of parameters while keeping most of the network frozen. These approaches significantly reduce

training demands but still require model updates and do not address the limitations of static knowledge.

A central challenge remains that fine-tuned models internalize information in their parameters, making it difficult to update or correct their knowledge without retraining. This is particularly problematic in domains where accuracy, traceability, and frequent updates are essential, such as technical documentation. These constraints motivate the exploration of approaches that can incorporate external knowledge without modifying model weights, which is the focus of the next section.

1.2.4 Applications and Limitations

LLMs have demonstrated remarkable versatility across a wide range of applications, including conversational agents, machine translation, text summarization, information retrieval, and code generation [37]. Their ability to generalize from examples enables few-shot and zero-shot learning, allowing them to perform tasks for which they have never been explicitly trained. In conversational systems, LLMs provide the generative and reasoning capabilities that support context-aware dialogue, intent understanding, and flexible interaction.

Despite these strengths, LLMs also present important limitations. Because they optimize for the likelihood of generating plausible text, they may produce incorrect or fabricated information, a phenomenon commonly referred to as hallucination. Their knowledge is bounded by the data used during pre-training, which makes it difficult to update or correct specific facts without retraining. Furthermore, inference with large models is computationally intensive, raising concerns related to scalability, environmental impact, and deployment cost. Additional challenges include biases inherited from training data, risks of misinformation, and privacy considerations when interacting with sensitive content.

These limitations are particularly relevant in domains where accuracy, traceability, and up-to-date information are essential. To address these issues, hybrid approaches such as Retrieval-Augmented Generation integrate external knowledge sources directly into the generation process and are discussed in the following section.

1.3 Retrieval-Augmented Generation

This section introduces Retrieval-Augmented Generation (RAG), a framework that extends LLMs by integrating external information retrieval into the generation process [38]. Instead of relying solely on the knowledge encoded in model parameters, RAG retrieves relevant evidence from an external corpus and incorporates it during inference. The following subsections outline the main concepts behind this approach, its advantages and challenges, and explain why RAG was particularly suitable for the system developed in this thesis.

1.3.1 Motivation and Background

LLMs rely on knowledge encoded in their parameters during pre-training, which limits their ability to incorporate new or domain-specific information. Updating this knowledge through fine-tuning is possible, but it is computationally expensive, requires labeled data, and offers limited flexibility when information must be refreshed frequently.

Retrieval-Augmented Generation offers an alternative approach by integrating an external retrieval mechanism into the generation process. Instead of relying solely on internal model parameters, RAG retrieves relevant information from an external corpus and uses it to guide the response generated by the language model. A schematic overview is presented in Figure 1.1.

The pipeline typically involves embedding the user query, searching for semantically similar vectors within a vector store, which is a specialized database that indexes documents as high-dimensional embeddings, and returning the most relevant passages. These retrieved contexts are then provided to the language model, enabling it to generate outputs grounded in explicit evidence.

The motivation behind RAG lies in extending LLMs beyond the constraints of a fixed training corpus while keeping inference efficient. This approach is particularly advantageous in scenarios where information evolves rapidly or where accuracy and traceability are essential, making it well suited for technical and domain-specific applications.

1.3.2 RAG: Advantages, Challenges, and Comparison with Fine-Tuning

Fine-tuning and Retrieval-Augmented Generation both extend the capabilities of pre-trained language models, but they do so in fundamentally different ways. Fine-

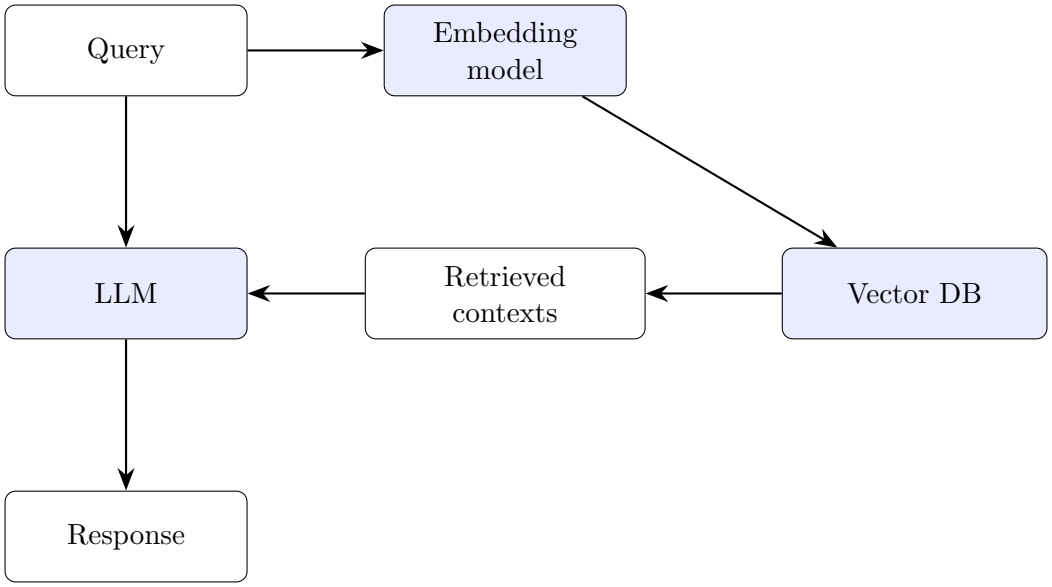


Figure 1.1: High-level diagram of a RAG pipeline.

tuning internalizes new information into the model parameters through gradient-based optimization, which makes it effective for stable and well-defined tasks but costly to update when data changes. RAG instead externalizes knowledge: relevant information is retrieved from an external corpus at inference time and used to condition the model's output. This separation between knowledge storage and generation enables flexible and lightweight adaptation to new domains without modifying model parameters.

RAG offers several practical advantages. By grounding responses in retrieved evidence, it improves factual accuracy and reduces the risk of hallucinations. Its modular design allows the underlying knowledge base or retrieval component to be updated independently of the language model, and the retrieved passages provide transparency by making supporting evidence directly inspectable. These properties make RAG suitable for scenarios that require up-to-date, domain-specific, or verifiable information.

In addition to these advantages, RAG offers important benefits from a data safety perspective. Because the retrieval component operates on a locally stored document corpus, sensitive or proprietary manuals can be indexed and queried without exposing their full contents to the language model provider. Only the user query and the retrieved text fragments need to be processed by the model, which reduces the amount of internal documentation transmitted outside the organization. This separation between the knowledge base and the model improves privacy, facilitates compliance with company

policies, and enables on-premise or hybrid deployments where the retrieval pipeline is fully controlled by the engineering team. Nevertheless, although grounding answers in retrieved evidence mitigates hallucinations, it does not eliminate them entirely; therefore, critical configuration decisions should always be validated by the engineer.

RAG also introduces specific challenges. Its effectiveness depends on retrieval quality: irrelevant or noisy results can degrade the final output. Current language models are constrained by finite context windows, which limit how much retrieved information can be incorporated at once. In addition, real-time retrieval may introduce latency, and maintaining large vector databases requires careful data management and indexing strategies.

For these reasons, RAG represents an effective compromise between adaptability, efficiency, and factual reliability. Its ability to combine the generative strengths of language models with the precision of retrieval-based methods makes it particularly suitable for the chatbot developed in this thesis, which must operate on technical and frequently updated documentation.

1.4 Prompt Engineering

The behavior of LLMs is strongly influenced by how the input prompt is formulated. Prompt engineering refers to the design of prompts that guide the model toward producing relevant, accurate, and context-aware responses. It has become an essential technique for adapting general-purpose models to specific tasks without additional training and it plays a complementary role to retrieval-based methods such as RAG [39].

1.4.1 Definition and Motivation

Although LLMs are capable of performing a wide range of linguistic and reasoning tasks, their outputs depend heavily on the clarity and structure of the provided input. Prompt engineering aims to reduce ambiguity, encourage appropriate reasoning, and improve factual reliability by shaping the model's internal processing through carefully designed instructions. This approach aligns with the broader paradigm of in-context learning, in which a model adapts its behavior based on the examples, constraints, or stylistic cues contained within the prompt itself.

1.4.2 Common Prompting Techniques

Several prompting strategies have been developed to improve model performance depending on the task:

- **Zero-shot prompting:** the model receives only a direct instruction. This is effective for well-defined tasks where the expected format is clear.
- **Few-shot prompting:** one or more input-output examples are included in the prompt to illustrate the desired behavior. This helps the model infer the task structure and response style.
- **Instruction-based prompting:** explicit instructions are added to shape the response, such as "Explain step by step" or "Summarize in short bullet points."
- **Chain-of-thought prompting:** the model is encouraged to generate intermediate reasoning steps before providing the final answer, which can improve performance on tasks that require structured reasoning.
- **Role-based prompting:** the model is assigned a specific persona or role (for example, "You are a technical assistant"), which influences tone, style, and focus. This technique is particularly useful in domain-specific applications.

Other prompting strategies exist, such as self-consistency prompting, which samples multiple reasoning paths and selects the most coherent one, although such methods are typically used in specialized settings.

Different prompting techniques can also be combined. For instance, a role-based prompt may be paired with explicit instructions and a short example, allowing the model to better understand both the expected behavior and the output format.

1.4.3 Challenges and Future Directions

Despite its utility, prompt engineering presents several challenges. Models can be highly sensitive to small changes in phrasing or structure, which may lead to variations in output quality. Long prompts consume more tokens, increasing computational cost and sometimes approaching context length limits. Moreover, evaluating prompt quality remains difficult, as no standard metrics exist beyond task-specific performance.

Recent research has explored automated methods such as prompt tuning or reinforcement-based optimization, in which prompts are refined through data-driven procedures rather than manual design. While these approaches aim to provide more systematic control over model behavior, prompt engineering remains a practical and widely used technique for guiding LLM outputs and complements retrieval-based approaches by further shaping how external knowledge is used during generation.

In the context of this thesis, prompt engineering is employed to steer the model's use of retrieved information, ensure consistency with the technical domain, and improve the overall quality of the chatbot's responses.

Chapter 2

Development and Implementation

This chapter describes the practical development of the chatbot system for interacting with microcontroller manuals. It presents the preliminary feasibility study, the data preparation and preprocessing pipeline, the tools and libraries adopted, the strategies for chunking and prompting, and the final integration of these components into a functional Retrieval-Augmented Generation (RAG) architecture.

2.1 Preliminary Research and Feasibility Study

Before developing a custom pipeline, a preliminary phase of exploratory research was carried out to assess whether existing no-code platforms could serve as a practical starting point for building a domain-specific chatbot. These platforms have become increasingly popular because they allow users to create AI assistants by uploading documents and configuring simple workflows, without requiring programming experience. Since this approach could be appealing for organizations interested in experimenting with AI solutions without introducing new technical expertise, it was useful to evaluate their capabilities and limitations in the context of microcontroller manuals. Although exploratory only and not part of the final system, this phase helped shape several choices later adopted in the implementation, including prompt design strategies and the selection of the large language models used.

MindStudio[1] was selected as the primary platform for this feasibility study due to its intuitive interface, workflow-oriented design, and free availability during the summer

[1] `https://mindstudio.ai/`

of 2024, when these experiments were conducted. The platform supports the creation of Retrieval-Augmented Generation (RAG) pipelines by automating key operations such as document ingestion, text chunking, embedding creation, and similarity search. This made it suitable for rapid prototyping and for exploring different configuration options without the need to develop a custom backend. An example of the workflow interface used during this phase is shown in Figure 2.1.

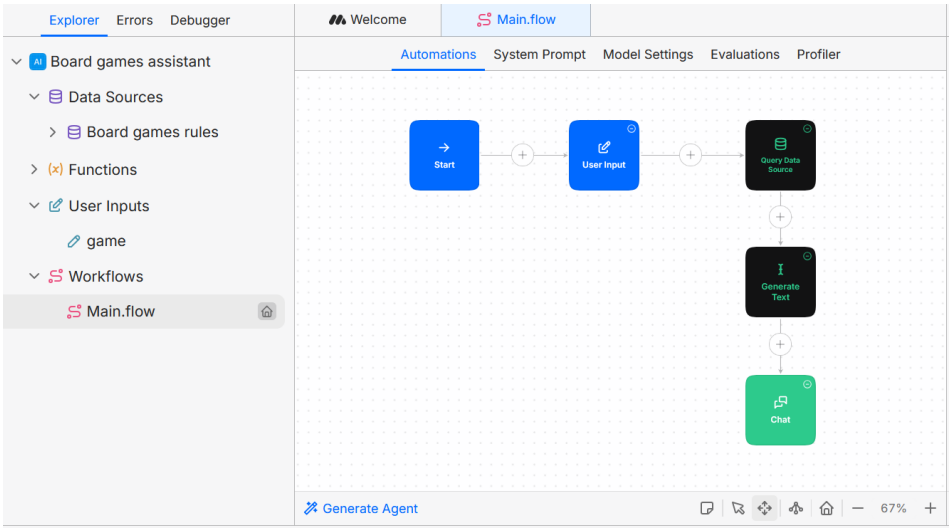


Figure 2.1: Workflow interface in MindStudio used during the feasibility study.

An initial challenge emerged immediately: evaluating the correctness of the chatbot's responses using microcontroller manuals was difficult without substantial domain expertise. These documents contain highly specialized and interdependent information, and an incorrect answer can have different degrees of "incorrectness": it may be completely wrong or inappropriate, partially correct but missing essential details, or factually correct yet incomplete in a way that does not affect the final configuration. Distinguishing between these cases requires technical knowledge, which made the manuals unsuitable as a first environment for systematically exploring model behavior and configuration choices. To obtain clearer and more immediate feedback on model behavior, the first tests were therefore conducted using board game manuals.

Board game manuals provided a simpler and more familiar environment for experimenting with no-code workflows. Their limited length and accessible content made it easier to evaluate the accuracy, completeness, and consistency of the answers across different prompts, LLMs, and platform settings. At the same time, they share structural

similarities with technical documentation, including tabular data, hierarchical rules, and references to specific cases that must be followed precisely. This made them a suitable proxy for exploring the capabilities of a RAG-based pipeline under controlled conditions, without the confounding effect of domain complexity.

The feasibility study produced several insights. As expected, models equipped with retrieval provided more accurate and grounded answers compared to generic chatbots without external documents. More importantly, the study highlighted clear limitations in the automated preprocessing performed by the platform. When PDFs were uploaded directly, the extraction stage occurred entirely inside MindStudio and could not be configured or inspected. Tables were often misinterpreted, with misaligned or missing rows and columns, and in some cases they were completely omitted during text conversion. Since tables often contain critical register descriptions and configuration parameters in microcontroller manuals, reliable extraction is essential, and these issues significantly motivated the development of a dedicated preprocessing pipeline in later stages.

Different prompting strategies were also evaluated during this phase. Role-based instructions, concise formulations, and prompts requiring explicit reference to the source consistently produced more stable and grounded outputs. These observations guided the design of the system prompt later used in the final implementation.

Several LLMs available in MindStudio were compared during the experiments, including GPT-3.5 Turbo, Gemini-1.5 Pro, Claude 3 Sonnet, and Mixtral 8×7B. When answers depended on a single, clearly identifiable rule or paragraph, the models behaved similarly. For queries requiring cross-referencing multiple sections, Gemini-Pro consistently produced the most reliable and conservative responses in this controlled test domain. At the time of experimentation (summer 2024), Gemini 1.5 Pro offered a favorable balance between quality, stability, and available free credits, which made it the most suitable choice for the subsequent development phase. It is worth noting that several of the models used during this exploratory phase are no longer available in the same form, as model releases evolve rapidly. However, the conclusions drawn from these experiments remain valid for the period in which the project was carried out, and the same considerations apply to the implementation and evaluation presented in the following chapters.

In summary, the feasibility study showed that while no-code platforms offer accessible and rapid experimentation, they lack the precision and control required for processing

complex technical manuals, particularly in the handling of tables and cross-references. The insights gained from this phase informed key design choices in the custom pipeline, including the selection of the LLM, the design of prompts, and the need for a transparent preprocessing stage, which are described in the next sections.

2.2 Data Preparation

This section describes the preparation of the documentation used to build the chatbot's knowledge base. The project relies on two microcontroller reference manuals, selected to represent the type of technical material commonly consulted by firmware developers. After introducing their main characteristics, the section describes the preprocessing steps required to transform the original PDF files into structured text suitable for retrieval and embedding.

2.2.1 Data Description

The data used for this project consist of two microcontroller reference manuals from different manufacturers. These documents were selected because they represent the type of technical material that firmware developers routinely consult, and they form the knowledge base on which the chatbot must operate. Both manuals contain detailed descriptions of device architecture, peripherals, memory organization, and configuration registers, expressed through a combination of text, tables, diagrams, and numerical specifications.

The first document is the **STM32G0B1xB/xC/xE** reference manual published by STMicroelectronics (revision 3, January 2022) [40]. It describes a family of 32-bit Arm® Cortex®-M0+ microcontrollers running at up to 64 MHz. With approximately 160 pages, it combines explanatory text with a moderate number of tables and figures. Its compact structure and manageable complexity made it suitable for the early development and testing of the preprocessing workflow.

The second document is the **Kinetis KE17Z/13Z/12Z** reference manual from NXP Semiconductors (revision 1, June 2021) [41]. It covers another family of 32-bit Arm Cortex-M0+ microcontrollers, operating at up to 72 MHz, and exceeds 1000 pages. Compared to the STM32 manual, it presents a much denser layout with a higher concentration of tables, register descriptions, and functional diagrams. This level of detail

made it useful for evaluating how well the pipeline scales to long and heterogeneous documents and for demonstrating that the preprocessing strategy can adapt to different documentation formats.

Both manuals were used in their entirety and were available only in PDF format, exhibiting common characteristics of industrial technical documentation. These include multi-column layouts, nested and wide tables, repeated headers and footers, and cross-references spread across multiple chapters. Such formatting complicates automatic extraction and requires careful processing to preserve the relationships between text and tables; For these reasons, the manuals offer realistic examples of the challenges encountered when preparing technical documents for downstream retrieval and question answering.

To give a visual sense of their structure and layout, Figure 2.2 shows an example page from each manual. The images are not meant to be fully readable, but to give an idea of the type of formatting, the information density, and the usage of tables across the two documents.

A summary of their main characteristics is shown in Table 2.1. The substantial differences in length, structure, and information density provided a useful basis for testing the robustness and adaptability of the preprocessing and retrieval methods developed in this thesis. It is worth noting that the density values may appear higher than what one might expect from a simple ratio of tables and figures per page. This is because both manuals contain many long tables that span multiple pages. As a result, density reflects not only the number of distinct tables but also their physical distribution across pages.

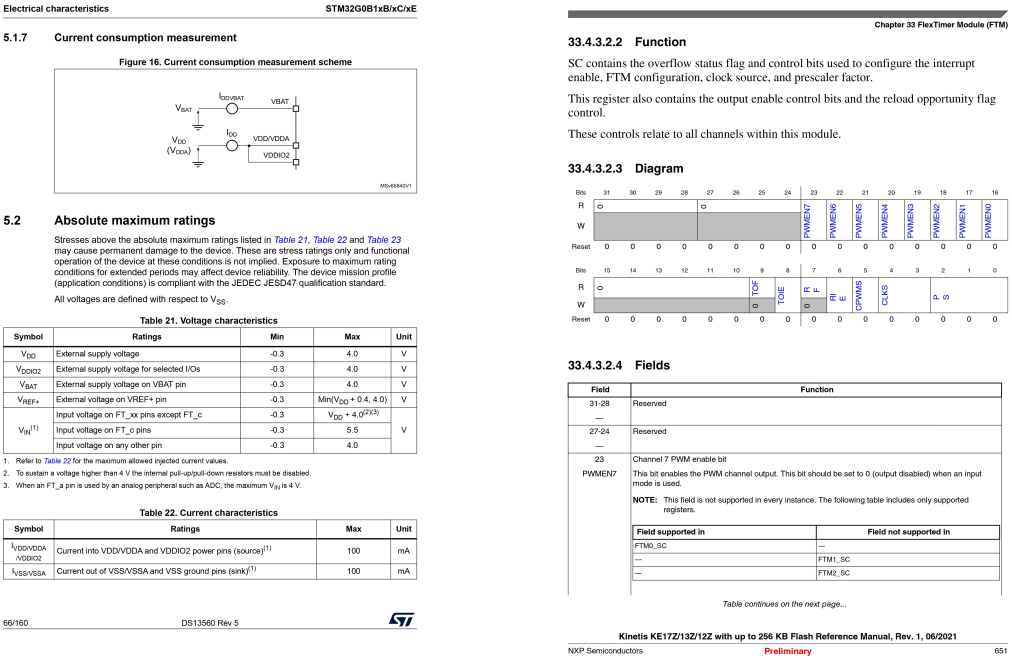
Manual	**Pages**	**Tables**	**Figures**	**Density***
STM32G0B1 series (ST)	160	96	66	1.28
Kinetis KE17Z series (NXP)	1103	~ 250	212	0.98

Table 2.1: Main characteristics of the reference manuals used in the project.

* Density values express the approximate ratio of tables and figures per page.

2.2.2 Data Preprocessing

Since the reference manuals were only available in PDF format, a dedicated preprocessing pipeline was developed to convert them into a structured and machine-readable



Electrical characteristics STM32G0B1xB/xC/xE

5.1.7 Current consumption measurement

Figure 16. Current consumption measurement scheme

5.2 Absolute maximum ratings

Stresses above the absolute maximum ratings listed in Table 21, Table 22 and Table 23 may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability. The device mission profile (application conditions) is compliant with the JEDEC JESD47 qualification standard.

All voltages are defined with respect to V_{SS}.

Table 21. Voltage characteristics

Symbol	Ratings	Min	Max	Unit
V_{DD}	External supply voltage	-0.3	4.0	V
V_{DDIO2}	External supply voltage for selected I/Os	-0.3	4.0	V
V_{BAT}	External supply voltage on VBAT pin	-0.3	4.0	V
V_{REF+}	External voltage on VREF+ pin	-0.3	Min(V_{DD} + 0.4, 4.0)	V
V_{IN}(1)	Input voltage on FT_xx pins except FT_c	-0.3	V_{DD} + 4.0(2)(3)	V
	Input voltage on FT_c pins	-0.3	5.5	
	Input voltage on any other pin	-0.3	4.0	

1. Refer to Table 22 for the maximum allowed injected current values.
2. To sustain a voltage higher than 4 V the internal pull-up/pull-down resistors must be disabled.
3. When an FT_a pin is used by an analog peripheral such as ADC, the maximum V_{IN} is 4 V.

Table 22. Current characteristics

Symbol	Ratings	Max	Unit
$I_{VDD/VDDA}$ /I_{VDDIO2}	Current into VDD/VDDA and VDDIO2 power pins (source)(1)	100	mA
$I_{VSS/VSSA}$	Current out of VSS/VSSA and VSS ground pins (sink)(1)	100	mA

66/160 DS13560 Rev 5

Chapter 33 FlexTimer Module (FTM)

33.4.3.2.2 Function

SC contains the overflow status flag and control bits used to configure the interrupt enable, FTM configuration, clock source, and prescaler factor.

This register also contains the output enable control bits and the reload opportunity flag control.

These controls relate to all channels within this module.

33.4.3.2.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				0				PWMEN7	PWMEN6	PWMEN5	PWMEN4	PWMEN3	PWMEN2	PWMEN1	PWMEN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						TOF	TOIE	RF	RIE	CPWMS	CLKS		PS		
W							0		0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

33.4.3.2.4 Fields

Field	Function
31-28 —	Reserved
27-24 —	Reserved
23 PWMEN7	Channel 7 PWM enable bit This bit enables the PWM channel output. This bit should be set to 0 (output disabled) when an input mode is used. **NOTE:** This field is not supported in every instance. The following table includes only supported registers.

Field supported in	Field not supported in
FTM0_SC	—
—	FTM1_SC
—	FTM2_SC

Table continues on the next page...

Kinetis KE17Z/13Z/12Z with up to 256 KB Flash Reference Manual, Rev. 1, 06/2021

NXP Semiconductors Preliminary 651

Figure 2.2: Example page layouts from the STM32G0B1 reference manual (left) and the Kinetis KE17Z reference manual (right).

corpus suitable for embedding and retrieval. The entire workflow was implemented in Python and designed to preserve the logical order of the documents while extracting both descriptive text and tabular information in a consistent format. Although preprocessing large manuals is computationally expensive, it must be performed only once per document, and the resulting corpus can be reused for all subsequent retrieval operations.

As a first step, each manual was automatically divided into chapters based on its table of contents. This segmentation reduced the computational load associated with processing large documents and simplified quality inspection by allowing extraction to be carried out and validated chapter by chapter.

The core of the preprocessing pipeline relied on a hybrid strategy combining two complementary tools. The `unstructured` library was used to parse each chapter and identify elements such as titles, paragraphs, tables, and figures, providing metadata for each extracted component. However, while it was effective at recognizing element types, its table extraction was not always reliable for the complex register tables commonly

found in microcontroller documentation. To obtain precise cell-level structures, the `Camelot` library was applied in lattice mode to extract grid-based tables. Camelot alone, however, occasionally misclassified figures or diagrams as tables. For this reason, the pipeline integrated both tools: `unstructured` indicated which elements were tables, while Camelot provided structured table content. Additional filtering removed spurious tables by discarding Camelot outputs with a high proportion of empty cells.

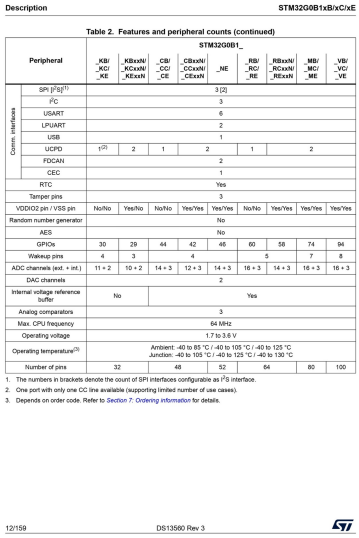
To illustrate the challenges involved in parsing complex technical tables, Figure 2.3 shows a representative page from the STM32G0B1 reference manual together with the structural elements identified by Camelot. The example highlights how the tool reconstructs ruling lines, localizes text regions, and combines these signals to infer the underlying table layout. In this particular case, the table contains multiple nested headers, irregular subcells, and non-uniform column groupings. Camelot is able to recover the structure with a high degree of accuracy, but certain fine-grained elements are not captured perfectly due to the complexity of the layout. Such cases demonstrate that, while automated extraction is generally reliable, highly irregular tables may still require additional inspection or manual correction during preprocessing.

After extraction, text and tables were recombined in reading order to reconstruct each chapter in a linear format. Tables were inserted as HTML-like representations to preserve their structure and cell relationships. This choice is consistent with recent research on applying LLMs to tabular data [42], where structured textual encodings are commonly used to make row and column relationships explicit for downstream models. Retaining this structure ensured that register tables were preserved faithfully and could be interpreted consistently within a text-based retrieval pipeline.

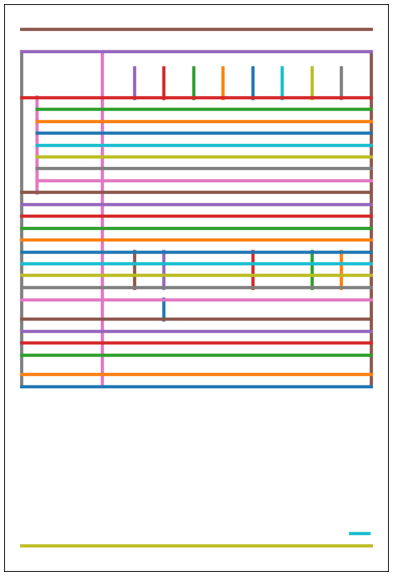
Relevant metadata such as chapter titles, table titles, and page markers was preserved to support traceability and facilitate later debugging. The final output consisted of a clean text representation for each chapter, containing both narrative content and structured tables. These chapter files were merged into a single corpus for each manual, forming the basis for the chunking and embedding procedures described in the next section.

2.2.3 Chunking and Vector Store Construction

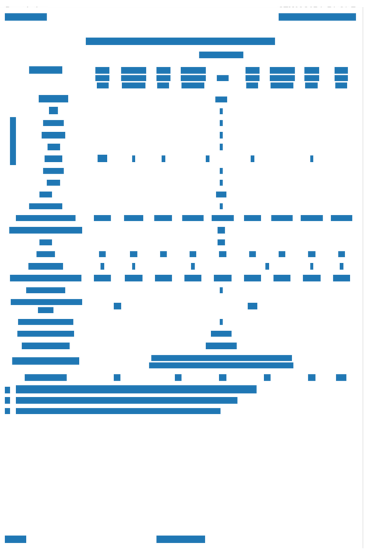
Once the manuals had been transformed into structured text files, the next step was to divide the corpus into smaller textual units and convert them into vector representa-



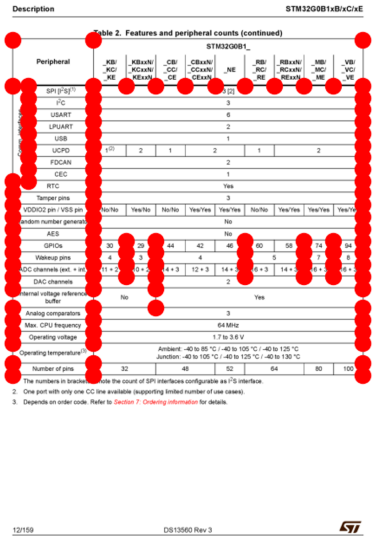
(a) Original page.



(b) Line detection.



(c) Text regions.



(d) Line intersections.

Figure 2.3: Example of a complex register table page and the corresponding structural interpretation produced by Camelot.

tions suitable for semantic retrieval. This operation, commonly referred to as *chunking*, determines the granularity at which information can be indexed and retrieved by the RAG system.

Because the manuals contain heterogeneous content—including descriptive text, HTML-

like representations of tables, parameter summaries, and configuration notes—special attention was required to avoid fragmenting coherent technical units. In particular, splitting register tables across chunks would significantly reduce retrieval quality. For this reason, an *adaptive* chunking strategy was adopted. Instead of using fixed-size segments or relying on section boundaries, the chunk size was computed dynamically as a function of the total document length, ensuring that each manual was divided into a comparable number of large chunks. A substantial overlap between consecutive chunks was maintained to preserve continuity and reduce the risk of breaking tables or separating related explanations. This approach provided a practical balance between retrieval accuracy and computational efficiency.

Each chunk was then encoded into a numerical vector using the `text-embedding-004` model from Google, which was selected for its robustness across heterogeneous chunk types and its consistent behavior when processing both textual descriptions and table-derived content.

All embeddings were stored in a vector database implemented with `FAISS` (Facebook AI Similarity Search), which enables efficient nearest-neighbor search even for large collections. For each user query, the system retrieves the top-5 most semantically similar chunks, which are then passed to the language model to support grounded answer generation. Metadata associated with each chunk, such as chapter identifiers and manual source, allows retrieved passages to be traced back to the corresponding sections in the original documents.

The entire chunking, embedding, and retrieval workflow was implemented using the `LangChain` framework, which provides a unified interface for text splitting, embedding creation, and vector store management. This modular structure ensures that individual components can be reconfigured or replaced if new tools become available.

The resulting knowledge base is a fully searchable semantic index of the manuals: every fragment of the documents is represented by its vector embedding, enabling the system to efficiently locate and combine relevant information during response generation in the RAG pipeline.

2.3 Prompt Engineering

Prompt design played an important role in controlling the chatbot's behavior and reducing hallucinations. In a RAG system, the prompt determines how the retrieved

passages are interpreted and integrated into the model's reasoning. While retrieval selects relevant content from the vector store, the prompt specifies how the model should use this content to produce coherent, evidence-based, and verifiable answers.

Several prompting strategies were tested during the experimentation phase to identify which formulations produced the most accurate and concise responses. Different styles were explored, including instruction-based prompts, role-based prompts, and variants that explicitly required the model to cite the source of each statement. The experiments showed that clear and directive formulations yielded more reliable outputs than generic instructions. In particular, assigning the model a specific role (for example, "expert assistant in firmware development") significantly improved factual accuracy by constraining the tone and scope of its responses.

The final configuration adopted a structured system prompt that defined the expected behavior of the assistant. The prompt instructed the model to act as a domain expert, to rely exclusively on the retrieved context, and to avoid speculation. An example is shown below:

> "You are an expert assistant in firmware development. Answer questions based strictly on the context retrieved from the manuals. If the information is not present, clearly state that it cannot be found, and reference the relevant section or page whenever possible."

This formulation encouraged concise and document-grounded responses while maintaining a cooperative conversational tone. By emphasizing transparency and explicitly discouraging unsupported inferences, the prompt helped reduce hallucinations and improved the reliability of the generated answers. The requirement to cite sections when possible also strengthened traceability, which is essential when interacting with technical documentation.

Overall, the prompt engineering phase established the behavioral constraints necessary for consistent and verifiable outputs. The resulting configuration integrates smoothly with the retrieval pipeline, ensuring that the model's responses remain aligned with the underlying documentation and suitable for the technical domain considered in this thesis.

2.4 System Integration

The components developed throughout the project were integrated into a unified conversational architecture that connects data preprocessing, retrieval, and answer generation. This integration ensures that each stage of the pipeline, from document ingestion to response delivery, operates coherently, enabling users to query technical manuals through natural language.

The system follows the typical structure of a Retrieval-Augmented Generation framework. When the user submits a question through the command-line interface (CLI), the query is embedded and compared against the vectors stored in the knowledge base. The retriever identifies the top five most semantically relevant chunks, which are then passed to the language model together with the user query and the system prompt. The LLM generates an answer grounded in these retrieved passages, and the final output is returned to the user together with references to the corresponding sections or pages of the manuals. This approach ensures that the generated responses remain both context-aware and traceable.

All system components were implemented using the LangChain framework, which offers a flexible environment for connecting preprocessing, retrieval, and generation modules. The modular structure makes it possible to configure or replace individual components independently, including the text splitter, embedding model, vector store, retriever, and language model. Conversation management is handled through LangChain's message-passing interface, which maintains dialogue history and allows interactions to be reused for inspection or later evaluation.

The chatbot currently operates through a CLI, a choice motivated by transparency and ease of debugging. This interface makes it possible to display retrieved passages alongside generated answers, helping to assess retrieval quality and identify hallucinations during iterative development. Although the system was prototyped in a text-based environment, the underlying architecture is general and can be extended to graphical or web-based interfaces without modification to the core logic.

Figure 2.4 illustrates the overall workflow of the system. The process begins with the ingestion of the reference manuals, which are transformed by the preprocessing pipeline into a structured corpus. These data are then chunked, embedded, and stored in the vector database. During interaction, the user query triggers the retriever, which selects the most relevant fragments to provide grounding for the LLM, resulting in a coherent

and verifiable answer.

The integrated system combines data engineering and language modeling components within a single modular framework. This design enables efficient interaction with complex technical documentation and provides firmware developers with accurate, context-aware, and traceable responses.

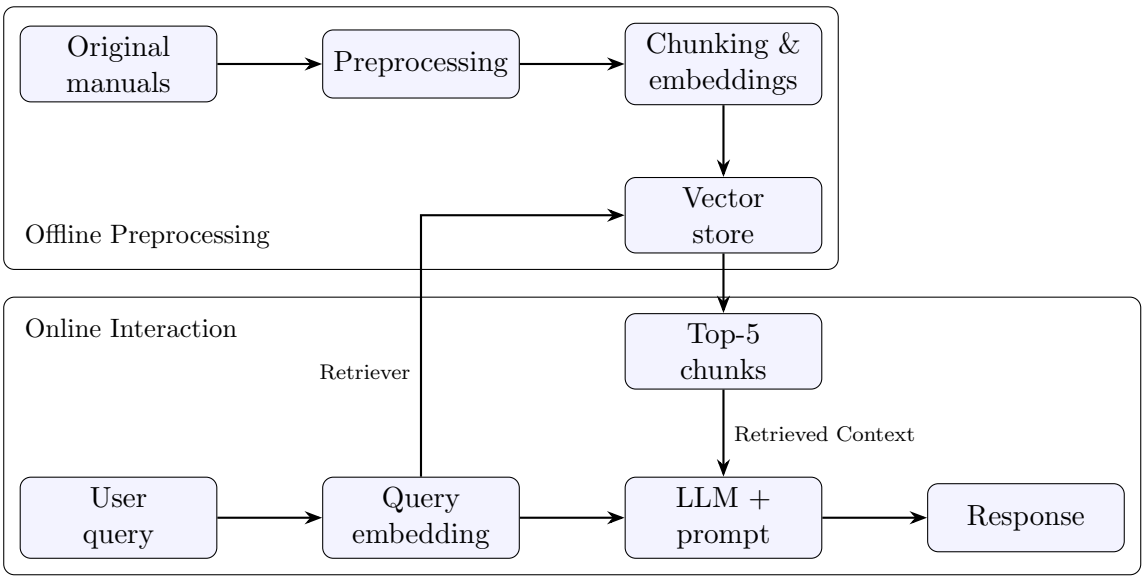


Figure 2.4: Overview of the chatbot architecture. Manuals are preprocessed offline to produce embeddings stored in a vector database. At query time, the user question is embedded, relevant chunks are retrieved from the vector store, and the language model generates a response based on the combined query and retrieved contexts.

Chapter 3

Evaluation and Analysis

This chapter presents the evaluation of the developed chatbot system, focusing on its accuracy, completeness, and practical usability when interacting with microcontroller reference manuals. Since no public benchmark exists for this specific domain, the assessment relied on a manually curated set of test questions prepared in collaboration with domain experts from Brain Technologies. Each response produced by the chatbot was compared against the official documentation and reviewed according to predefined qualitative criteria. The chapter describes the evaluation methodology, summarizes the obtained results, examines representative examples, and discusses the main behavioral patterns and practical considerations that emerged from the experiments.

3.1 Evaluation Methodology and Metrics

The objective of the evaluation was to assess how accurately and reliably the developed chatbot can assist engineers in consulting microcontroller reference manuals. Since the experiments were tailored to the specific documentation, a dedicated set of test questions was manually created in collaboration with domain experts. These questions reflect realistic information needs encountered during firmware development, including requests about pin functions, electrical limits, register fields, and peripheral configuration steps.

Automatic text similarity metrics such as BLEU or ROUGE were considered, but ultimately found unsuitable for this context. Such metrics assume the availability of large datasets with multiple reference answers per question, enabling meaningful statistical comparisons. In this project, the expected responses are short, technical statements that

can be phrased correctly in several different ways, making similarity-based metrics a poor indicator of factual accuracy. Constructing a sufficiently large domain-specific dataset with validated alternative answers would have required extensive expert annotation and was outside the scope of this work.

For these reasons, the evaluation relied on qualitative, human-interpretable criteria. Each chatbot response was rated according to the following metrics:

- **Correctness**: the degree to which the answer matches the information contained in the manual. Ratings were assigned on a five-point scale (1 = incorrect, 5 = fully correct).
- **Completeness**: the extent to which the answer includes all relevant details needed to fully address the question. Again, ratings were manually assigned on a scale from 1 to 5.
- **Citation accuracy**: whether the answer provides a correct reference to a relevant table, figure, section, or page of the manual.
- **Clarity and relevance**: a qualitative judgment of how understandable, concise, and practically useful the answer is for an engineer.

All responses were reviewed by domain experts, who cross-checked them against the official documentation. Because this process requires detailed verification of specifications and descriptions in the documentation, the size of the test set represents a practical balance between representativeness and feasibility.

Alongside these metrics, additional aspects such as topic coverage and general response time were also considered as part of the overall evaluation, providing complementary insight into the system's usability and robustness.

The following sections present the test set, summarize the results, and examine representative examples.

3.2 Test Set and Experimental Setup

The evaluation was conducted on the two microcontroller reference manuals introduced in Section 2.2.1. A set of 24 test questions was manually defined in collaboration

with domain experts, reflecting realistic information needs encountered during the consultation of technical documentation.

The test set was organized into two complementary groups. The first group consisted of questions whose answers correspond to precise, explicitly stated information in the manuals, typically retrievable from tables, parameter listings, or short descriptive paragraphs. These questions were intended to assess the chatbot's ability to locate and extract well-defined technical details.

The second group required combining information from multiple sections of the documentation. These questions involved multi-step reasoning or the integration of related descriptions across different chapters, providing a more challenging evaluation scenario. Although smaller in number, this second group offered insight into the system's ability to navigate dependencies and interpret more complex relationships within the manuals.

Each response was evaluated according to the criteria described in Section 3.1. While the size of the test set was necessarily limited due to the manual validation required for each case, the selected questions cover a representative variety of lookup-oriented and reasoning-oriented tasks. As such, they provide a realistic indication of the system's expected behavior during practical interactions with technical documentation.

3.3 Quantitative Results

Table 3.1 summarizes the main quantitative outcomes of the evaluation. For each manual, the average correctness and completeness scores reflect the qualitative ratings described in Section 3.1, while citation accuracy indicates the proportion of answers that included an explicit and correct reference to the relevant part of the documentation.

Manual	N. Questions	Correctness (avg)	Completeness (avg)	Citation. Acc.
STM32G0B1	19	4.0/5	4.1/5	0.79
KE17Z	5	4.4/5	4.8/5	0.80

Table 3.1: Summary of evaluation results.

The correctness score reflects how accurately each answer matched the information contained in the manual. A score of 5 corresponds to a fully correct response, while a score of 4 indicates generally correct information with minor imprecisions. Lower values represent increasing levels of inaccuracy or missing details. Answers that were incorrect

on the first attempt but corrected immediately after a follow-up prompt were assigned an intermediate score of 2.5. This approach captures both the initial error and the model's ability to revise its output when prompted, highlighting the interactive nature of the system. A similar strategy was adopted for completeness scoring, where answers that initially omitted relevant details but recovered them during follow-up interaction were reflected in intermediate values.

The decision to assign intermediate scores to answers that were incorrect on the first attempt but corrected after a follow-up prompt reflects the intended usage scenario of the system. In practical settings, engineers often possess enough domain knowledge to immediately recognize an incorrect statement and ask the chatbot to re-check its reasoning. In such cases, the model's ability to revise its output becomes relevant to the overall user experience. However, since not all users may detect an erroneous first answer, the initial mistake cannot be ignored in the evaluation. The intermediate score therefore balances the chatbot's capacity for self-correction with the practical implications of producing an incorrect response at first glance.

The citation accuracy metric reports the fraction of answers that included a correct reference to a specific table, section, or page, supporting traceability during technical consultation. Although not quantified numerically, the clarity and relevance of the responses were also considered qualitatively when interpreting the results.

Overall, the high correctness and completeness averages for both manuals indicate that the chatbot was able to extract factual information reliably, even when the underlying documentation varied considerably in structure and complexity. The results also show that, although a few answers were incorrect on the first attempt, the conversational loop often enabled the model to adjust and improve its output when prompted, reinforcing the practical value of interactive guidance when using large language models in technical domains.

3.4 Representative Examples and Observed Behaviors

This section presents a selection of representative examples illustrating the chatbot's behavior during evaluation. Each example includes the question asked, the expected answer when applicable, and the response produced by the chatbot. A brief comment follows each case, highlighting the relevant aspects of the observed behavior.

Example A (STM32G0B1): Fully correct and complete answer

Question: Would the device be damaged if I have an input voltage higher than 4 V on a `FT_xx` pin?

Expected answer: No, if the internal pull-up or pull-down resistor is disabled; yes otherwise.

Chatbot answer: "While the absolute maximum rating for input voltage on a `FT_xx` pin is VDD + 4.0 V (page 66), the manual explicitly warns: 'To sustain a voltage higher than 4 V, the internal pull-up/pull-down resistors must be disabled.' (Page 66, Table 21 footnote 2) Therefore, exceeding 4 V without disabling the internal resistors could potentially damage the device. It's crucial to carefully manage the input voltage and resistor configuration to ensure safe operation."

Comment: The chatbot retrieves the relevant warning from the correct table footnote and provides a complete and technically accurate explanation.

Example B (STM32G0B1): Partially correct answer

Question: What pin should I connect to to take `VREFBUF`?

Expected answer: `VREF+` (Table 12).

Chatbot answer: "Unfortunately, based on the provided manual alone, I cannot definitively tell you which pin to connect to for `VREFBUF`. e manual mentions `VREF+` several times but does not explicitly state its pin location for all package options. [...]
Here's what I found:
Page 17: Explains that `VREF+` is delivered through the `VREF+` pin. However, it also states, 'On packages without `VREF+` pin, `VREF+` is internally connected with VDD, and the internal voltage reference buffer must be kept disabled (refer to datasheets for package pinout description). [...]' "

Comment: The chatbot extracts the correct conceptual information and explicitly signals its uncertainty. Although it does not cite the specific table, it still identifies the correct pin and provides a valid rationale. This behavior is particularly valuable in

practical usage: by clearly indicating its level of confidence while pointing to relevant sections of the manual, the chatbot supports informed verification and enables the user to continue the interaction with appropriate awareness.

Example C (STM32G0B1): Incorrect answer even after a second interaction

Question: Can pin `PA3` be used as `TIM15_CH2`?

Expected answer: Yes.

Chatbot answer (first attempt): "No, `PA3` cannot be used as `TIM15_CH2`. Table 13 (Page 56): This table displays the alternate function mappings for Port A, and it does not list `TIM15_CH2` as an option for `PA3` under any of the listed alternate functions (`AF0` to `AF7`)."

Follow-up dialogue: `User:` "Can you check again?"
`Chatbot:` "[...] I re-examined the manual and the relevant tables, and I confirm that:
`PA3` cannot be used as `TIM15_CH2`. Neither Table 13 nor Table 14 lists `TIM15_CH2` as an alternate function for `PA3`. [...]"

Comment: The chatbot cites the correct tables but misinterprets their content. Even after prompting, it repeats the same incorrect conclusion, illustrating a stable misreading of table entries rather than a hallucination or retrieval failure.

Example D (KE17Z): Multi-step reasoning across manual sections

First question: I need to drive a buzzer: it needs two symmetrical square-wave signals. How do I do that?

Chatbot answer: "The provided manual excerpt does not explain how to drive a buzzer and doesn't mention if it's possible to generate two symmetrical square-wave signals. Therefore, I cannot answer your question with the information I have."

Refined question: I need to generate two symmetrical signals, meaning that the output of one channel must be the inverse of the output of the other channel. How can I implement this?

Chatbot answer: "You can achieve this by using the Complementary mode.
The Complementary mode is selected when:
- `DECAPEN` = 0
- `COMBINE` = 1
- `COMP(m)` = 1 (where `m` represents a channel pair)
Page 737, Chapter 33 FlexTimer Module (FTM)
This mode uses a pair of channels (`n`) and (`n+1`), where the output of channel (`n+1`) is the inverse of the output of channel (`n`).
You will need to configure the appropriate registers to enable this mode and set the desired PWM characteristics for your application."

Comment: Faced with an over-generic first query, the chatbot responds conservatively. When the question is refined, it immediately retrieves the correct configuration mechanism and provides a detailed, accurate explanation drawn from multiple sections of the manual.

3.5 Observed Limitations and Practical Considerations

The evaluation revealed several recurring patterns in the chatbot's behavior, which help characterize both its strengths and its current limitations. Overall, the system was consistently reliable when responding to factual queries whose answers were located in short descriptive passages or clearly structured parameter listings. In these cases,

retrieval was effective, and the generated responses were accurate, concise, and often accompanied by correct citations.

The most frequent source of inaccuracy involved the interpretation of complex tables. The preprocessing pipeline successfully converted the majority of tabular structures into a usable textual representation, but highly irregular or deeply nested tables remain challenging to treat automatically. In these cases, the chatbot occasionally retrieved the correct table but focused on the wrong entry, leading to incomplete or incorrect conclusions. These errors stem from the inherent difficulty of navigating dense, multi-column layouts rather than from unsupported fabrication. When processing shorter or less complex documents, this issue was noticeably reduced, suggesting that table density and structural complexity are significant factors influencing retrieval accuracy.

A second pattern concerned the stability of responses across longer conversational sessions. As the dialogue progressed, the chatbot sometimes became more verbose, occasionally drifting from the prompting constraints defined at the start of the interaction. In particular, later responses tended to omit citations that were correctly provided earlier in the same session. Reinserting the system prompt or keeping interactions short proved effective in mitigating these effects.

Despite these issues, interaction itself remained beneficial. When explicitly prompted to re-check or reconsider an answer, the chatbot could often correct misinterpretations, especially for cases where the first attempt resulted from reading the wrong table or section. This interactive refinement reflects an important aspect of practical use, since engineers naturally iterate on their understanding when consulting technical documentation.

From a usability standpoint, the system proved helpful in reducing the effort required to locate information within long manuals. The ability to phrase questions in natural language, refine them incrementally, and obtain context-aware explanations contributes to a smoother consultation workflow compared to manual search alone. This was particularly evident in scenarios involving multi-step reasoning or cross-referencing between different sections of a document.

Depending on the confidentiality of the documentation, organizations may also need to consider how the system is deployed. In cases where manuals contain proprietary information, it may be preferable to keep document embeddings and the vector store on-premise while relying on a cloud-based model only for inference. Such hybrid setups

allow companies to benefit from high-quality language models while maintaining control over sensitive data. Practical considerations also include the cost of using commercial LLMs, which typically follow a token-based pricing model. While occasional use incurs negligible expense, sustained or large-scale deployment may require strategies such as prompt optimization, caching, or the selective use of smaller local models to balance accuracy with operational cost.

In summary, the system performs reliably in factual retrieval tasks and provides meaningful support for navigating technical documentation. Its main limitations are associated with dense table interpretation and long-session consistency, especially when working with large or highly structured manuals. Despite these challenges, the overall behavior is robust enough to assist engineers effectively during early design exploration and routine consultation tasks.

Conclusions

This thesis investigated the development of a domain-specific chatbot based on Retrieval-Augmented Generation (RAG), designed to support engineers in consulting microcontroller reference manuals. The project, carried out in collaboration with Brain Technologies, examined how large language models can contribute to technical documentation workflows and evaluated their reliability in this context.

The work combined several components into a coherent prototype system. A preprocessing pipeline was developed to convert heterogeneous PDF manuals into a structured and searchable knowledge base, preserving both descriptive text and tabular information. The chatbot was then built to interpret user queries and generate answers grounded in the retrieved documentation. The overall objective was to assess whether such a system could help engineers navigate complex reference material more efficiently.

The evaluation results indicate that the proposed approach is effective for factual retrieval and short reasoning tasks. The chatbot reliably identified technical values, configuration details, and parameter constraints across different types of manuals. Its conservative prompting strategy contributed to stable and document-grounded responses. At the same time, the experiments highlighted clear limitations, particularly in the interpretation of dense or irregular tables and in maintaining prompt consistency during long interactions. These findings reflect the challenges inherent in processing technical documentation and the current capabilities of retrieval-based systems.

From a practical perspective, the prototype already offers useful support. By allowing engineers to pose questions in natural language and receive concise source-based answers, it reduces the effort needed to locate information dispersed across long manuals. However, a deployment-ready version would require additional features such as the ability to store and manage larger collections of documents, support for regularly updated data, and at least a minimal user interface to integrate the system into existing

engineering workflows.

Several directions for future development emerge from this work. Improving the handling of complex tables, and extending preprocessing to include figures, diagrams, and layout cues, could substantially broaden the system's coverage. Alternative strategies such as converting structured information into dedicated databases or exploring multimodal models may also improve precision for tabular and graphical content. On the modeling side, techniques for adaptive prompt reinforcement or the use of lightweight local models may increase reliability and reduce operational cost. Extending the pipeline to additional document types and designing user interfaces tailored to engineering workflows would further enhance its practical impact.

More broadly, this thesis was completed in a period of rapid progress in large language models. Capabilities that were experimental only a few years ago have become widely accessible, and models continue to improve in context handling, reasoning, and multimodal understanding. In this evolving landscape, the approach explored here represents one of many promising ways to integrate language models into technical tasks, and future advancements are likely to expand what such systems can accomplish.

In conclusion, the project showed that RAG-based chatbots can meaningfully assist in navigating complex technical documentation. While limitations remain, the results suggest that such systems can improve information accessibility, reduce manual search time, and provide a practical starting point for more advanced intelligent documentation tools in embedded systems engineering.

Bibliography

[1] Oxford University Press. "chatbot (n.)". https://doi.org/10.1093/OED/2981785869, 2023. Accessed: 2025-11-15.

[2] Bayan Abu Shawar and Eric Atwell. Chatbots: are they really useful? *Journal for Language Technology and Computational Linguistics*, 22(1):29–49, 2007.

[3] Kiran Ramesh, Surya Ravishankaran, Abhishek Joshi, and K Chandrasekaran. A survey of design techniques for conversational agents. In *International conference on information, communication and computing technology*, pages 336–350. Springer, 2017.

[4] Madeleine Bates. Models of natural language understanding. *Proceedings of the National Academy of Sciences*, 92(22):9977–9982, 1995.

[5] Stevan Harnad. Language writ large: Llms, chatgpt, grounding, meaning and understanding. *arXiv preprint arXiv:2402.02243*, 2024.

[6] Erin Sanu, T Keerthi Amudaa, Prasiddha Bhat, Guduru Dinesh, Apoorva Uday Kumar Chate, and Ramakanth Kumar. Limitations of large language models. In *2024 8th International Conference on Computational System and Information Technology for Sustainable Solutions (CSITSS)*, pages 1–6. IEEE, 2024.

[7] Alan M Turing. Computing machinery and intelligence. In *Parsing the Turing test: Philosophical and methodological issues in the quest for the thinking computer*, pages 23–65. Springer, 2007.

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

[9] Kenneth Mark Colby, Sylvia Weber, and Franklin Dennis Hilf. Artificial paranoia. *Artificial intelligence*, 2(1):1–25, 1971.

[10] Eleni Adamopoulou and Lefteris Moussiades. Chatbots: History, technology, and applications. *Machine Learning with applications*, 2:100006, 2020.

[11] Richard Wallace. The elements of aiml style. *Alice AI Foundation*, 139:35, 2003.

[12] Richard S Wallace. The anatomy of alice. In *Parsing the Turing test: Philosophical and methodological issues in the quest for the thinking computer*, pages 181–210. Springer, 2007.

[13] Sumit Kumar Dam, Choong Seon Hong, Yu Qiao, and Chaoning Zhang. A complete survey on llm-based ai chatbots. *arXiv preprint arXiv:2406.16937*, 2024.

[14] D. A. Ferrucci. Introduction to "this is watson". *IBM Journal of Research and Development*, 56(3.4):1:1–1:15, 2012.

[15] Matthew B Hoy. Alexa, siri, cortana, and more: an introduction to voice assistants. *Medical reference services quarterly*, 37(1):81–88, 2018.

[16] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and machines*, 30(4):681–694, 2020.

[17] John Schulman, Barret Zoph, Christina Kim, Jacob Hilton, Jacob Menick, Jiayi Weng, Juan Felipe Ceron Uribe, Liam Fedus, Luke Metz, Michael Pokorny, et al. Chatgpt: Optimizing language models for dialogue. *OpenAI blog*, 2(4), 2022.

[18] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[19] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[20] Oliver Knill, Johnny Carlsson, Andrew Chi, and Mark Lezama. An artificial intelligence experiment in college math education. Published online, 2004. Available at `http://www.math.harvard.edu/knill/preprints/sofia.pdf`.

[21] Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. The impact of ai on developer productivity: Evidence from github copilot. *arXiv preprint arXiv:2302.06590*, 2023.

[22] Luminița Nicolescu and Monica Teodora Tudorache. Human-computer interaction in customer service: the experience with ai chatbots—a systematic literature review. *Electronics*, 11(10):1579, 2022.

[23] Zhi Wei Lim, Krithi Pushpanathan, Samantha Min Er Yew, Yien Lai, Chen-Hsin Sun, Janice Sing Harn Lam, David Ziyou Chen, Jocelyn Hui Lin Goh, Marcus Chun Jin Tan, Bin Sheng, et al. Benchmarking large language models' performances for myopia care: a comparative analysis of chatgpt-3.5, chatgpt-4.0, and google bard. *EBioMedicine*, 95, 2023.

[24] Manoj Kumar Kamila and Sahil Singh Jasrotia. Ethical issues in the development of artificial intelligence: recognizing the risks. *International Journal of Ethics and Systems*, 41(1):45–63, 2025.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[26] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[27] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[29] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[32] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 7871–7880, 2020.

[33] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[34] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 13484–13508, 2023.

[35] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[36] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[37] Rishi Bommasani. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[38] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024.

[39] Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. Unleashing the potential of prompt engineering for large language models. *Patterns*, 2025.

[40] STMicroelectronics. *STM32G0B1xB/C/E Reference Manual*, revision 3 edition, January 2022. RM0444.

[41] NXP Semiconductors. *KE1xZK Series Microcontrollers Reference Manual*, revision 1 edition, June 2021. Document Number: KE1xZRM.

[42] Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, and Christos Faloutsos. Large language models (llms) on tabular data: Prediction, generation, and understanding– a survey. *arXiv preprint arXiv:2402.17944*, 2024.