Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження моделей створення чат-ботів, що імітують стиль спілкування людини
(тема)

Виконав:
здобувач другого року навчання,
групи СШІм-23-1
Іван Колесніков
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва освітньої програми)
Керівник доц. Олександра Вітько
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 25 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Колеснікову Івану Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження моделей створення чат-ботів, що імітують стиль спілкування людини

затверджена наказом університету від 21 квітня 20 25 р. № 295Ст

2. Термін подання студентом роботи до екзаменаційної комісії 4 червня 20 25 р.

3. Вихідні дані до роботи дані Інтернет-джерел, наукові публікації, Python документація

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі
2) Вивчення механізмів пам'яті великих мовних моделей
3) Порівняння та вибір великої мовної моделі
4) Імплементація архітектури великих мовних моделей
5) Проведення експериментальних досліджень
6) Опис результатів експериментальних досліджень
7) Формування висновків з результатів експериментальних досліджень

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	21.04.2025	виконано
2	Ознайомлення з технічним завданням	23.04.2025	виконано
3	Аналіз предметної галузі	25.04.2025	виконано
4	Вибір бібліотек Python для реалізації механізмів	30.04.2025	виконано
	пам’яті		
5	Імплементація конфігурацій LLM	01.05.2025	виконано
6	Проведення експериментальних досліджень	05.05.2025	виконано
7	Інтерпретація результатів досліджень	14.05.2025	виконано
8	Написання пояснювальної записки	19.05.2025	виконано
9	Перевірка на академічний плагіат	22.05.2025	виконано
10	Нормоконтроль	27.05.2025	виконано
11	Підготовка презентації та доповіді	28.05.2025	виконано
12	Попередній захист	29.05.2025	виконано
13	Рецензування	30.05.2025	виконано
14	Захист перед ЕК	04.06.2025	

Дата видачі завдання 21 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Олександра Вітько
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 58 с., 9 рис., 9 табл., 2 дод., 23 джерела.

ВЕЛИКІ МОВНІ МОДЕЛІ, ГЕНЕРАЦІЯ ЗА ДОПОМОГОЮ ІНФОРМАЦІЙНОГО ПОШУКУ, ДІАЛОГ, КОНТЕКСТ, ЧАТ-БОТИ.

Мета дослідження – покращення імітації людської розмови у чат-ботах шляхом підвищення контекстної обізнаності, зв’язності та адаптивності через модифікацію архітектури та розширення механізмів пам’яті.

Об’єкт дослідження – використання та порівняння різних стратегій пам’яті в системах чат-ботів.

Предмет дослідження – розробка гнучкої системи прийняття рішень для вибору оптимальної архітектури чат-бота залежно від вимог до діалогу, таких як довжина контексту, якість відповіді та здатність до збереження пам’яті.

Методи дослідження – поєднання теоретичного аналізу та експериментальної перевірки. Теоретичний етап охоплював огляд літератури щодо сучасних архітектур чат-ботів та механізмів отримання інформації. На експериментальному етапі було розроблено та протестовано декілька конфігурацій чат-ботів у контрольованих умовах.

Результати експериментів показали, що моделі з фіксованим вікном контексту є найбільш збалансованими незалежно від довжини контексту. Системи з узагальненням пам’яті загалом мали найгірші результати, але покращувались із розширенням контексту. Архітектури RAG досягли найкращих показників при довгому контексті. Температура 0.5 забезпечувала оптимальну якість відповідей. На основі цих даних розроблено систему вибору архітектури відповідно до вимог діалогу та ресурсів.

ABSTRACT

Master’s thesis contains: 58 pp, 9 fig., 9 tabl., 2 ann., 23 references.

CHATBOTS, CONTEXT, CONVERSATION, LARGE LANGUAGE MODELS, RETRIEVAL-AUGMENTED GENERATION.

Objective of research – improving the imitation of human conversation in chatbots by enhancing contextual awareness, coherence, and adaptability through architectural modifications and memory augmentation.

Subject of research – utilizing and comparing different memory strategies within chatbot systems, including fixed-context larg language models (LLM), summary-based memory buffers, and Retrieval-Augmented Generation (RAG) pipelines to evaluate their impact on conversation quality.

Purpose of research – to develop a flexible decision-making framework for selecting the most suitable chatbot architecture based on dialogue requirements such as context length, response quality, and memory persistence.

Methods of research – this study combines theoretical analysis and experimental validation. The theoretical phase involved a comprehensive review of literature on modern chatbot architectures, LLM capabilities, and retrieval mechanisms. The experimental phase included building and experimenting with multiple chatbot configurations under controlled conditions.

The experiment results revealed that LLMs using a fixed context window were the most well-rounded performers across all context sizes. Summary-based systems performed the weakest overall but improved as context length increased. RAG systems showed the highest performance in extended context scenarios, particularly at 18,000 tokens. Temperature also played a critical role, with 0.5 yielding the most balanced scores across models. Based on these findings, a new architecture selection framework was proposed to guide developers in configuring chatbots according to their specific performance and resource requirements.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

LLM – Large Language Model;

LTM – Long-Term Memory;

RAG – Retrieval-Augmented Generation;

BERT – Bidirectional Encoder Representations From Transformers;

GPT – Generative Pre-Trained Transformer.

INTRODUCTION

In recent years, large language models have become central to the development of intelligent conversational agents, offering increasingly human-like dialogue capabilities across a wide range of applications. From customer support and personal assistants to educational tools and creative writing aids, the demand for chatbots that can simulate coherent, context-aware, and realistic human interaction continues to grow. While the core architecture of modern chatbots often relies on transformer-based LLMs, the strategies used to manage memory and integrate contextual information play a critical role in determining overall conversational quality.

This research investigates how different architectural configurations, namely fixed context windows, summarization-based memory compression, and retrieval-augmented generation, influence a chatbot's ability to imitate human conversation in multi-turn settings. The study evaluates each approach along key dimensions of performance, including precision, recall, F1 score, and semantic similarity (BERTScore), while also assessing the impact of parameters such as context length and temperature on dialogue quality.

The central aim of this work is to establish a practical and evidence-based framework for selecting appropriate chatbot architectures based on the specific demands of a use case. Through extensive experimenting and comparative analysis, the research provides insights into the trade-offs between simplicity, contextual depth, accuracy, and scalability. It demonstrates that no single configuration is universally optimal, but that strategic choices, guided by context length, required recall fidelity, and performance constraints, can significantly improve chatbot utility and realism.

In doing so, this thesis contributes to the broader understanding of how memory and retrieval mechanisms shape conversational behavior and proposes a decision-making model to aid practitioners in deploying more effective and context-appropriate chatbot systems.

1 THEORETICAL OVERVIEW OF USAGE AI SYSTEMS FOR CREATING CHATBOTS AND DEFINING PURPOSE OF THE RESEARCH

In today's digital landscape, chatbots have become integral to human-computer interactions, facilitating automated communication across various domains, from customer service to educational platforms. The ability of chatbots to emulate human conversational styles is crucial for enhancing their effectiveness and user acceptance.

1.1 Chatbots history

The concept of a chatbot – a system capable of simulating human conversation – has evolved significantly over time. According to the Cambridge Dictionary, a chatbot is «a computer program designed to have a conversation with a human being, usually over the internet» [1]. While this definition describes modern implementations, the roots of the idea go much deeper.

The origin of chatbots can be traced back to Alan Turing's influential 1950 paper «Computing Machinery and Intelligence» where he posed the question: «Can machines think?» [2]. Turing proposed what is now known as the Turing Test – a method of evaluating a machine's ability to exhibit intelligent behavior indistinguishable from that of a human. This foundational idea introduced the notion that a machine could potentially mimic human conversation.

In 1966, Joseph Weizenbaum at MIT created ELIZA, widely considered the first working chatbot. ELIZA simulated a Rogerian psychotherapist by using pattern-matching and substitution to respond to user input [3]. Although primitive by today's standards, ELIZA gave users the illusion of understanding, laying the groundwork for future developments in natural language processing. The example of conversation with ELIZA is shown in figure 1.1.

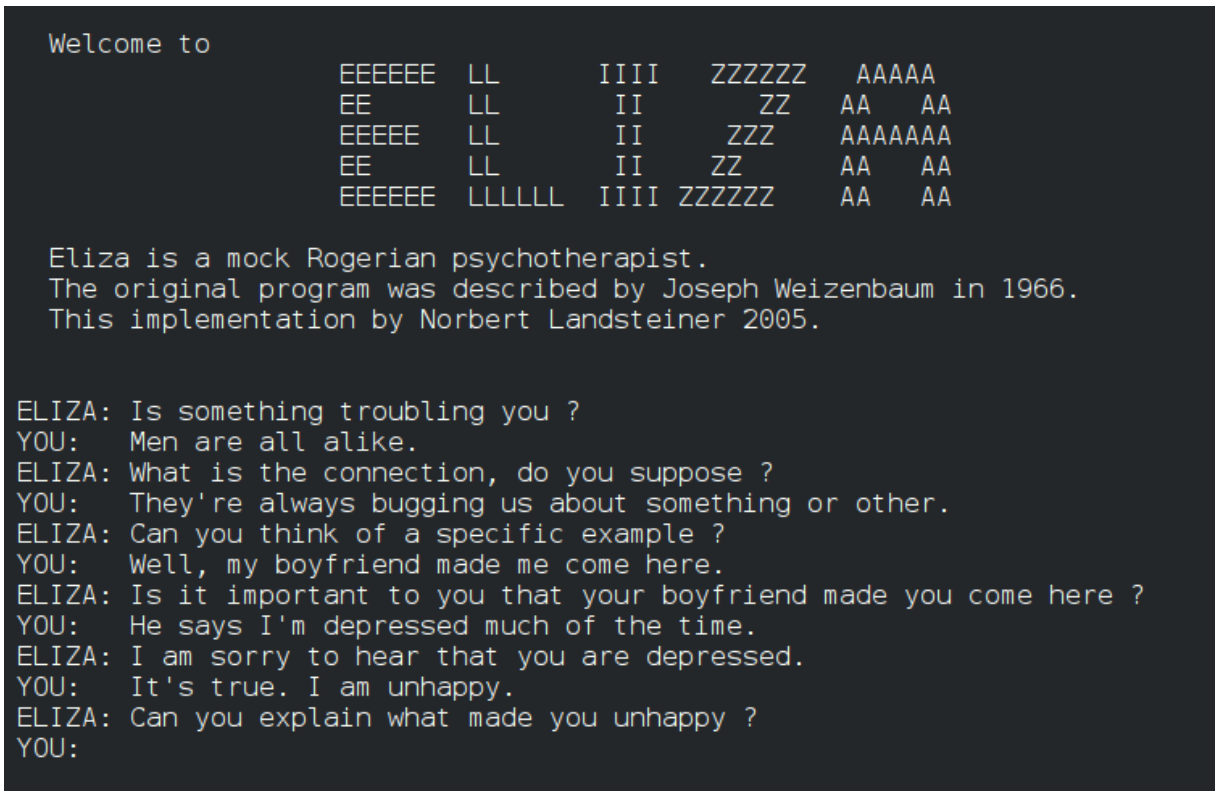

Figure 1.1 – Conversation with ELIZA

Following ELIZA, several rule-based systems emerged, such as PARRY (1972), a chatbot designed to simulate a person with paranoid schizophrenia. Unlike ELIZA, PARRY included a model of emotions and beliefs, representing a more complex and psychologically-informed approach. Later, systems like ALICE (Artificial Linguistic Internet Computer Entity, 1995) and Mitsuku continued to refine rule-based conversation, winning various Loebner Prizes for their attempts to pass the Turing Test.

However, until the 2010s, chatbot development was constrained by limitations in computing power, data availability, and algorithmic complexity. Most bots followed pre-scripted rules and failed to maintain meaningful long-term context.

The introduction of machine learning, particularly deep learning, brought a major turning point. Chatbots were no longer bound to hand-coded rules and instead began to learn from large datasets of real human interactions. This shift

enabled a more flexible and scalable approach to dialogue systems, culminating in the development of transformer-based models – an architectural breakthrough introduced by Vaswani et al. in 2017 [4].

In the last decade, chatbots powered by transformers and LLMs (e.g., OpenAI's ChatGPT, Google's Gemini, Anthropic's Claude, and Meta's Llama) have significantly advanced the field. These systems are capable of maintaining context across longer conversations, generating coherent replies, and even adapting to the user's tone and intent. Figure 1.2 illustrates the increasing use of the term «chatbot» in written sources between 2000 and 2022, reflecting growing public and academic interest.

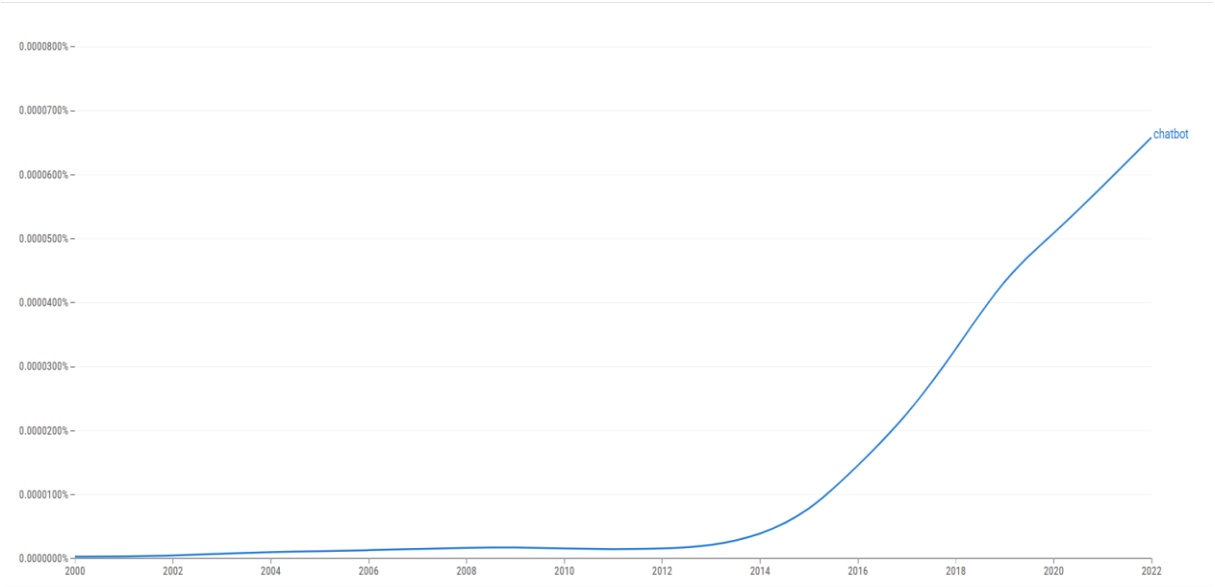

Figure 1.2 – Frequency of the word «chatbot» found in written sources from 2000 to 2022 years

1.2 Current state of chatbots

The development of chatbots has accelerated rapidly over the past decade, largely due to the availability of large-scale datasets, increased computational power, and advances in natural language understanding. Modern chatbots range

from narrow-purpose assistants to open-domain conversational agents capable of engaging in general dialogue.

Chatbots systems can be broadly categorized into:

– menu or button-based chatbots, which are the most basic kind of chatbots and serve mostly as a customer support tool to answer repetitive, straightforward questions;

– rule-based chatbots, which still serve in constrained environments due to their predictability and control. They function according to conditional «if-else» logic and keyword detection. These types of chatbots can be easily trained and excel at answering common questions, but quality of the answers quickly deteriorates in case of more complex queries;

– AI-powered chatbots, which can understand user's queries despite the phrasing. Such chatbots by means of machine learning can self-learn, keep the memory of previous interactions with users, thus providing more intelligent and meaningful interactions;

– voice chatbots, which are AI-chatbots that utilize text-to-speech or speech-to-text technology allowing users to speed up getting results by talking to the bot and eliminating the need for typing and submitting queries;

– generative chatbots, particularly those based on large language models, which can produce new, contextually relevant text in real-time;

– hybrid chatbots, which combine different types of chatbots simultaneously, providing the best user experience in complex situations [5].

The shift towards LLM-driven chatbots – such as OpenAI's ChatGPT, Google's Gemini, and Claude by Anthropic – has redefined expectations for what a chatbot can do. These models utilize the transformer architecture and are trained on massive datasets, enabling them to understand nuance, context, and even emulate human conversational style.

Additionally, hybrid approaches such as Retrieval-Augmented Generation combine the generalization ability of LLMs with domain-specific accuracy by

integrating external knowledge sources. These methods offer improved factual consistency and are gaining traction in enterprise and academic contexts.

1.3 State-of-art LLM architectures

Transformer models have revolutionized the field of artificial intelligence by introducing a novel architecture that excels at processing sequential data. Unlike traditional recurrent neural networks , transformers utilize a self-attention mechanism, allowing them to consider all elements of an input sequence simultaneously. This parallel processing capability enables the models to capture long-range dependencies more effectively and significantly reduces training time.

In NLP, transformers have become the backbone of large language models such as BERT and GPT. BERT employs an encoder-decoder structure to understand the context of words in a sentence, enhancing tasks like text classification and question answering [6]. On the other hand, GPT models utilize a decoder-only approach, excelling in text generation tasks by predicting the next word in a sequence [7].

Beyond NLP, transformer models have been adapted for various domains, including computer vision and speech recognition. Vision transformers apply the transformer architecture to image data, often outperforming convolutional neural networks in tasks like image classification and object detection [8]. Similarly, in speech recognition, transformers have been employed to improve the accuracy and efficiency of models.

1.4 Purpose of the research

This research investigates how different architectural configurations of large language models influence a chatbot’s ability to simulate human-like conversation in both short and extended contexts. Specifically, the study focuses on evaluating and comparing three primary system types:

– an LLM using only a fixed context window (LLM + Context);

– an LLM enhanced by summarization to preserve past dialogue (LLM + Summary);

– an LLM integrated with a retrieval-augmented generation pipeline (LLM + RAG).

The central hypothesis is that memory handling and retrieval mechanisms play a critical role in shaping the factual accuracy, coherence, and realism of chatbot responses. To test this, the study systematically measured performance across different context lengths and temperature values using BERTScore and custom evaluation logic to penalize non-informative outputs.

The research aims to:

– identify performance trade-offs between simplicity, factual precision, and conversational depth;

– propose a data-driven decision-making framework for selecting optimal chatbot architecture based on application needs;

– establish temperature tuning strategies that balance response creativity and factual correctness.

By consolidating these findings, the study offers a practical and scalable methodology for developing more effective, context-aware conversational agents tailored to specific interaction scenarios.

2 MEMORY AND RETRIEVAL IN LLM-BASED CHATBOTS

2.1 Memory mechanisms in conversational AI

One of the defining challenges in building human-like conversational agents is managing memory – i.e., the ability to maintain continuity, recall past interactions, and build on previous context. Different memory strategies offer varying trade-offs between performance, scalability, and realism.

2.1.1 Context-only memory

Context-only systems rely entirely on the immediate dialogue history passed within the model's context window. This approach is typical of basic LLMs, such as standard API calls to OpenAI's GPT-3.5 or GPT-4. These models generate responses based on the tokens included in the prompt, which usually contains the most recent parts of the conversation.

While simple to implement and cost-effective, this method has significant limitations:

– context truncation: as conversations grow, older messages must be dropped to fit within the maximum token limit;

– lack of long-term memory: the model cannot recall information shared earlier in the session or across sessions;

– no personalization or continuity beyond what fits in the prompt.

Despite its shortcomings, context-only memory remains popular due to its simplicity and the strong generative capabilities of modern LLMs.

2.1.2 Short-term memory using summarization

Summarization serves as an effective mechanism for maintaining conversational continuity in systems constrained by limited context windows.

Instead of storing full interaction histories, a summarization-based approach periodically condenses previous dialogue turns into a compact form, which is then appended to subsequent prompts. This technique allows the model to retain essential information while reducing token consumption, thereby extending the effective memory capacity.

Summarization is particularly advantageous in scenarios involving moderately long conversations, where memory depth is needed but cannot rely on expensive retrieval or long context windows. While this method enables efficient use of available context space, it introduces certain limitations. The quality of the summary directly affects downstream model behavior – important nuances or user-specific intents may be lost or misrepresented.

2.1.3 Long-term memory with vector stores using RAG framework

In the pursuit of creating conversational agents that emulate human-like interactions, the integration of long-term memory mechanisms has become paramount. Unlike short-term memory, which retains information temporarily, long term memory (LTM) enables AI agents to store and recall information across different sessions, facilitating personalization and contextual continuity over extended periods.

The workflow of LTM consists of several key components. First, it involves embedding generation, where textual or structured data is transformed into vector representations using embedding models. These embeddings are then stored and managed in a vector storage system, often utilizing vector databases such as FAISS or Chroma. To find relevant information, similarity retrieval is employed, using metrics like cosine similarity to retrieve pertinent vectors in response to user queries. Finally, contextual integration incorporates this retrieved information into the LLM’s input, which helps to inform and enhance the generation of responses.

The incorporation of LTM offers several advantages: personalization, as it retains user-specific information, allowing AI agents to tailor interactions to individual users, enhancing user experience and engagement; contextual continuity, which enables agents to maintain context over multiple sessions, facilitating more coherent and meaningful conversations; knowledge retention, permitting agents to accumulate knowledge over time, improving their ability to provide accurate and relevant information; and adaptability, which facilitates the agent's ability to adapt to new information and evolving user needs without retraining the underlying model [9].

2.2 Retrieval using RAG framework

Having context as a limiting factor of modern LLMs, the problem of models not being able to provide accurate and meaningful information has arised. Retrieval-Augmented Generation addresses these issues by dynamically retrieving pertinent data from external repositories, thereby grounding the model's outputs in current and verifiable information. Retrieval-Augmented Generation is an advanced AI framework designed to enhance the capabilities of large language models by integrating external knowledge sources during the response generation process [9].

The RAG framework (figure 2.1) operates through a two-step process:

– retrieval: upon receiving a user query, the system searches external databases or documents to fetch relevant information. This ensures that the model has access to the most recent and contextually appropriate data;

– generation: the retrieved information is then combined with the original query and fed into the LLM, which generates a response that is both contextually enriched and grounded in the latest data.

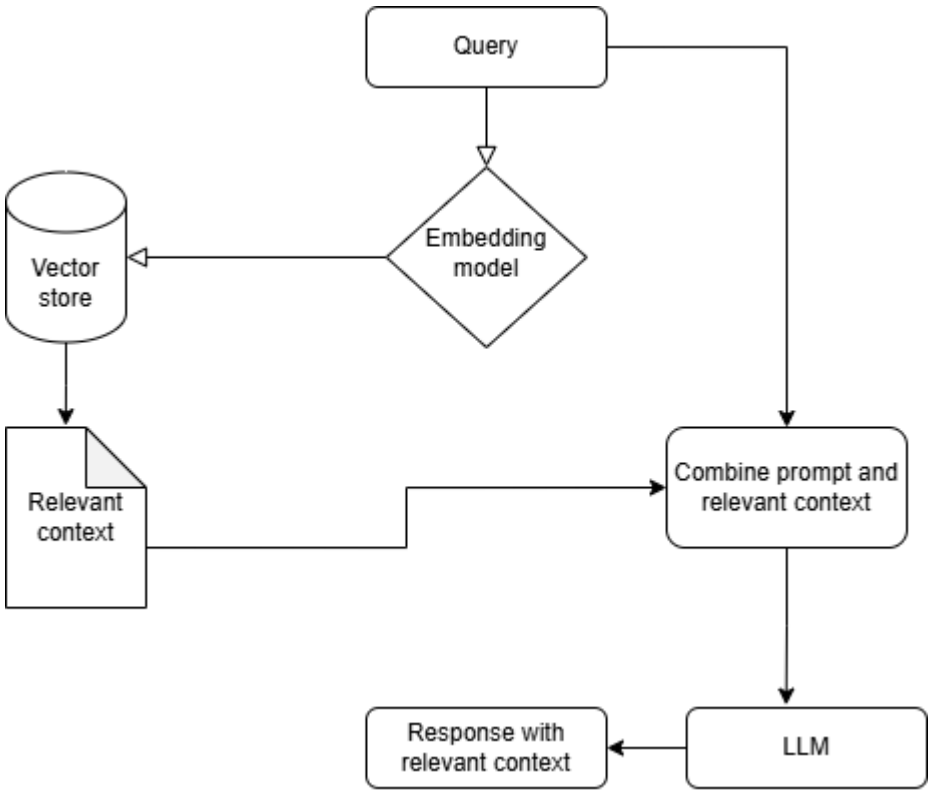

Figure 2.1 – Visualization of RAG framework

This approach effectively transforms the LLM's operation from a «closed-book» to an «open-book» model, allowing it to reference external sources akin to consulting a textbook during an exam [10].

RAG framework excels in enhancing accuracy by leveraging up-to-date external information, reducing the likelihood of generating incorrect or «hallucinated» responses, thereby improving the reliability of AI outputs. Also it mitigates the inherent limitations of an LLM's context window by incorporating extensive external data, enabling the model to handle complex queries that require broader contextual understanding. Lastly, the framework allows for seamless integration of new information without necessitating retraining of the model, making it particularly valuable in dynamic fields where knowledge evolves rapidly.

2.3 Types of RAG architectures

RAG systems can be categorized based on their complexity and the sophistication of their retrieval and generation processes:

– naive RAG: this basic implementation involves a straightforward retrieval of information based on the user's query, which is then fed into the language model to generate a response. While simple and quick to deploy, naive RAG lacks optimization mechanisms, making it less suitable for complex applications requiring high accuracy;

– advanced RAG: building upon the naive model, advanced RAG incorporates sophisticated algorithms such as rerankers, fine-tuned LLMs, and feedback loops. These enhancements improve accuracy, adaptability, and performance, making advanced RAG more suitable for complex, real-world applications;

– modular RAG: this architecture emphasizes a modular design, allowing for flexibility and scalability. By structuring the retrieval and generation components as interchangeable modules, modular RAG facilitates easier updates and customization, catering to specific application needs [11].

Each of these architectures offers distinct advantages and can be selected based on the specific requirements of the application, such as the need for speed, accuracy, or scalability.

2.4 Embedding-based search

At the core of the RAG framework lies the concept of embedding-based search – a technique that enables the retrieval of semantically relevant documents based on vector similarity rather than keyword overlap. Unlike traditional search engines that rely on lexical matching, embedding-based search captures the underlying meaning of text, making it more suitable for complex, natural language queries.

The process begins by transforming both the user query and the textual documents in the knowledge base into high-dimensional vectors. These vector representations, or embeddings, are generated using pretrained transformer-based models, such as BERT, Word2Vec, or similar encoder architectures [12]. These models are trained to place semantically similar texts close to one another in vector space, even if they do not share the same vocabulary.

For example, the queries «How to treat a sore throat?», «Best remedies for throat pain?» may be lexically distinct but will be mapped to nearby points in vector space, ensuring that relevant results are retrieved even when keywords do not match directly (figure 2.2).

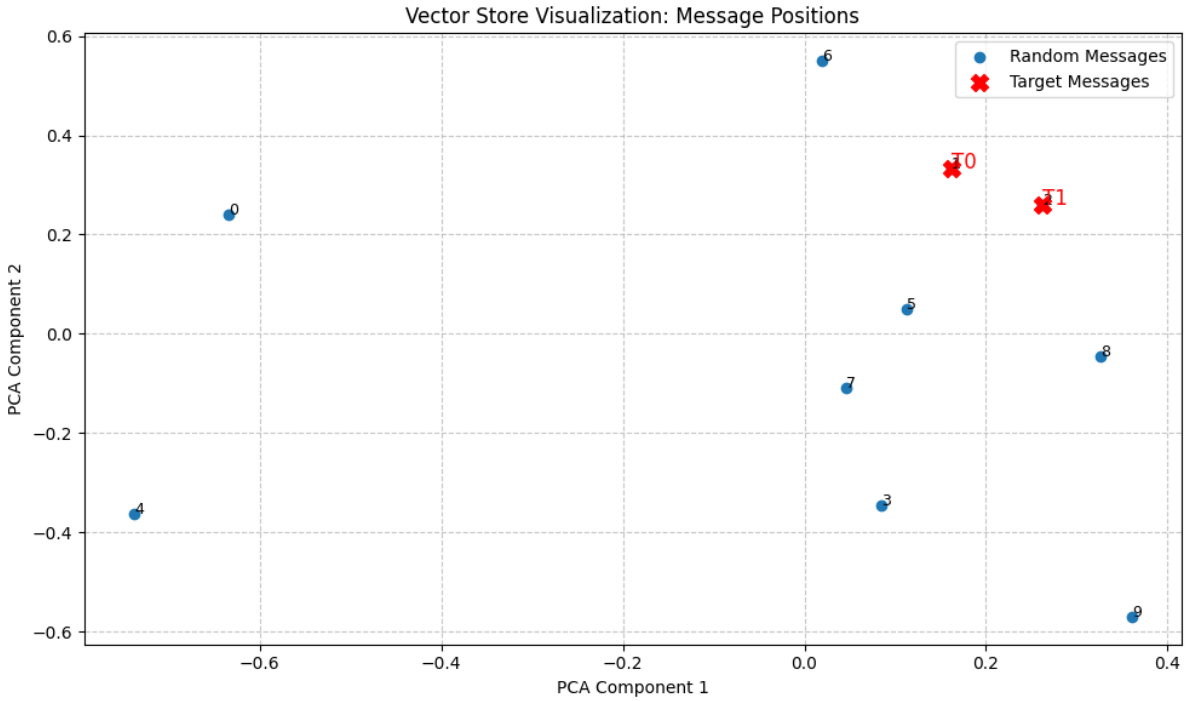

Figure 2.2 – Visualization of the mapping of «How to treat a sore throat?», «Best remedies for throat pain?» messages in the vector store

Once embedded, the query vector is compared to all document vectors in a vector database using a similarity metric – most commonly cosine similarity or Euclidean distance. The top-k most similar documents are selected and passed on

to the generation phase. This approach allows for highly flexible, semantically-aware retrieval and supports multilingual, paraphrased, or domain-specific queries.

Key characteristics:

– approximate nearest neighbor search: used to scale retrieval across large document corpora while maintaining performance [13];

– chunking and indexing: long documents are typically split into (chunks) to improve retrieval granularity and relevance;

– libraries and tools: popular frameworks include FAISS, Chroma and Pinecone.

Embedding-based search provides multiple significant benefits. The technology understands the intended meaning behind queries rather than relying on matching specific keywords. It effectively processes alternative phrasings, similar terms, and even text with minor spelling errors. When properly configured, this approach works across different languages and specialized fields. The result is information retrieval that better aligns with the user’s actual intent, producing more coherent and precise responses.

In the context of a RAG pipeline, embedding-based search enables the dynamic retrieval of knowledge fragments from large, unstructured corpora. These retrieved documents are appended to the original query, forming an enriched context input for the language model.

3 LLM SELECTION AND ITS PARAMETERS

3.1 LLM comparison and choice argumentation

The development and evaluation of chatbots that imitate human communication necessitate the selection of an appropriate large language model. This section presents a comparative analysis of several state-of-the-art LLMs available as of 2025, considering factors such as model size, context window, modality support, and API availability (table 3.1). These criteria are crucial for tasks involving coherent dialogue, understanding user intent, and dynamic conversation management.

Table 3.1 – Comparing of LLMs

Model	Provider	Parameters	Context window	API availabi-lity	Notable features
1	2	3	4	5	6
Gemini 2.0 Flash [14]	Google DeepMind	Undisclosed	1M+ tokens	Yes (Vertex AI)	Optimized for speed and cost-efficiency, fast streaming responses
GPT-4.1 [15]	OpenAI	Undisclosed	1M tokens	Yes	Enhanced coding and instruction-following capabilities
Claude 3.5 [16]	Anthropic	Undisclosed	200K tokens	Yes	Improved performance on coding and multistep tasks, strong safety alignment

Table 3.1 continued

1	2	3	4	5	6
Mistral Large 2 [17]	Mistral	123B	128K tokens	Yes	Open-weight model with strong multilingual performance
Mixtral 8x7B [18]	Mistral	46.7B total (12.9B active)	32K tokens	Yes	Sparse Mixture of Experts architecture, efficient performance
Llama 3.1 [19]	Meta	8B / 70B / 405B	128K tokens	Yes (via third parties)	Open-source, customizable models with advanced safety features

Following this comparison, several critical factors inform the choice of model for this research. First, latency and cost are vital considerations in real-time chatbot applications. Gemini 2.0 Flash is explicitly designed to deliver low-latency outputs at a reduced computational cost, making it especially suitable for iterative experimentation and user-facing systems.

Second, the size of the context window determines how much conversational history or prior knowledge the model can retain and reason over. While Claude 3.5 and Mistral models offer solid mid-range capacity, Gemini 2.0 Flash and GPT-4.1 support up to one million tokens, enabling them to manage significantly longer and more complex dialogues.

Although the current thesis centers on text-based interaction, support for multimodal inputs such as images and audio has been considered for extensibility. Models like GPT-4.1 and Gemini 2.0 Flash already provide native vision

capabilities and could be adapted in future work involving visual chat or context-aware agents.

Given these considerations, Gemini 2.0 Flash was chosen due to its balanced performance profile. Its support for fast, streaming interactions allows for real-time chatbot responsiveness. Its extended context window enables more persistent and coherent dialogue. And finally, its availability through a robust and scalable API that works out of the box makes it a great choice for current research.

While larger or more capable models such as GPT-4.1 or Claude 3.5 may provide marginal gains in reasoning or creative generation, they also introduce higher latency, greater cost, and more integration complexity. In contrast, Gemini 2.0 Flash achieves a favorable balance between capability and efficiency, aligning closely with the objectives of this study: to explore how large language models can emulate human conversation in a responsive, accessible, and context-aware manner.

3.2 Consideration of local deployment: Llama 3.1

An additional direction considered in this research was the use of locally hosted models, particularly the open-source Llama 3.1 family released by Meta. These models, available in parameter sizes of 8 billion, 70 billion, and 405 billion.

From a functional standpoint, Llama 3.1 demonstrates strong performance across a range of natural language understanding and generation tasks. The 8B and 70B variants, in particular, offer competitive accuracy and reasoning abilities compared to commercial closed-weight models, especially when used with retrieval augmentation or optimized prompting. With context windows reaching up to 128,000 tokens, these models are capable of maintaining extended conversational history, which is critical for chatbot systems emulating human-like interaction.

Despite these advantages, Llama 3.1 was not selected for use in this work due to practical limitations associated with local deployment. The available

hardware for this study included a system equipped with an AMD Ryzen 5 2600 processor, 16 GB of RAM, and an AMD Radeon RX 580 graphics card. This configuration, while adequate for general development, was insufficient for the real-time inference demands of large-scale transformer models such as Llama 3.1.

During initial testing, the 8B model showed significant execution latency, particularly when running on CPU. Generation time for a single response often exceeded several seconds and came close to a minute, which made interactive conversation impractical.

Given the high inference latency, the local deployment of Llama 3.1 using Ollama did not meet the responsiveness requirements necessary for the intended real-time chatbot interface.

3.3 Fine-tuning selected model

In addition to model selection, the behavior of a language model during generation can be fine-tuned using decoding parameters such as temperature, top_k, and top_p. These parameters influence the randomness and diversity of the generated output. The temperature controls the sharpness of the probability distribution: lower values (e.g., 0.2–0.5) make the model more deterministic and focused, while higher values (e.g., 0.8–1.2) promote diversity and creativity. The top_k parameter limits generation to the top k most likely tokens at each step, filtering out the rest. This constrains output to a fixed subset of candidates and is often used to ensure more predictable generation. In contrast, top_p (nucleus sampling) includes the smallest set of tokens whose cumulative probability exceeds p, dynamically adjusting the sampling pool based on context. It allows for more flexible yet coherent output, especially when top_k is not used.

4 IMPLEMENTATION OF CHATBOT ARCHITECTURES

4.1 System architecture overview

This section provides a high-level overview of the three chatbot architectures evaluated in this study. Each architecture uses large language models to process user input and generate responses. The systems differ primarily in how they manage context and access external knowledge:

– LLM with context: the context is passed along with the user message in the same query;

– LLM with summary: summarization of previous conversations is done before passing it to the query along with the user's message;

– LLM with RAG framework: combination of a conversational agent and retrieval-augmented generation, allowing it to access an external knowledge base with previous conversation data and incorporate retrieved content into its responses.

Each system uses the Gemini Flash 2.0 LLM via API access. Retrieval functionality in RAG-based systems is implemented using vector similarity search through Chroma. Prompt construction, retrieval flow, and memory management are handled via modular pipelines using LangChain or LangGraph.

4.1.1 LLM with context

The conversational logic and evaluation process were implemented using the LangGraph framework, which provides a structured way to manage interactions with large language models in a stateful and modular fashion.

The core of the implementation is a simple graph where each node represents a functional step in the chatbot's workflow. In this case, a single node is responsible for sending user messages to the model and receiving its responses. The graph is initialized with an entry point that triggers this model invocation.

For each query, a predefined conversational context is combined with a user question, and this input is passed into the model through the graph. The system streams the model's output token by token, simulating a real-time chatbot interaction. This streaming mechanism also allows for low-latency feedback, which is beneficial in practical dialogue systems.

4.2.2 LLM with summarization

As another option, a pipeline with memory was implemented. This pipeline first summarizes the entire conversation and then allows for precise follow-up questioning based on that summary. The implementation uses LangChain.

The process begins with a summarization module implemented via an LLMChain, which uses a custom prompt to instruct the model to extract key facts, questions, answers, and conclusions from the previous conversation, if the number of messages exceeds 4. The use of a low temperature (0.1) ensures the summary remains focused and avoids creative deviation from the source.

Once the summary is generated, the system can continue the conversation. This is done using another prompt chain that guides the model to answer strictly based on the summary. The response generation again uses a low temperature to ensure clarity and factual accuracy. If the required information is not present in the summary, the model is instructed to state so clearly.

The prompt design plays a crucial role in getting the complete and clear summary. It's important to state the rules of how the LLM should summarize the text. The prompt (listing 4.1) emphasizes extracting key facts, relevant questions and answers, specific details like names or dates, and any final decisions or conclusions. The language in the prompt discourages general or vague paraphrasing and prioritizes factual content.

Listing 4.1 – Summarization query

```
Summarize the following conversation concisely, focusing on:
1. Key facts and information shared
2. Main questions asked and their answers
3. Specific details that might be referenced later (names, numbers, dates)
4. Any decisions or conclusions reached
Prioritize factual information over general discussion.
{conversation}
Summary:"""
```

4.2.3 LLM with RAG framework

To improve answer accuracy and factual grounding, especially when dealing with long or information-rich inputs, RAG pipeline was implemented. The approach combines vector-based semantic retrieval with generative answering by a large language model.

The RAG system is structured as a modular Python class. It initializes with a set of configurable parameters, including the LLM model, chunk size for document splitting, and temperature settings for response generation. A local embedding model (Llama3.2) is used to convert text into dense vector representations. These vectors are later stored in the Chroma vector database to enable semantic search.

Incoming documents or conversation histories are first segmented into overlapping chunks using a RecursiveCharacterTextSplitter. This ensures that each chunk preserves context. The resulting chunks are embedded and indexed into the vector store, making them retrievable by similarity.

When a user poses a question, the system first retrieves the most relevant document chunks using Maximum Marginal Relevance, which balances relevance and diversity in retrieval. This mechanism helps avoid redundancy while maintaining topic coverage. The selected content is then passed into a

retrieval-specific prompt, which explicitly instructs the language model to generate answers only based on the given context. If the necessary information is not available, the model is encouraged to state that clearly rather than speculate.

This architecture allows the chatbot to handle more complex queries by grounding its answers in verified source material. As a result, it significantly reduces hallucinations, increases factual correctness, and enhances the overall reliability of the system's outputs.

5 EXPERIMENTAL METHODOLOGY AND METRICS

Evaluating chatbot models that aim to imitate human communication presents a number of challenges, particularly in the selection of appropriate metrics and datasets. Most widely used evaluation metrics – such as BLEU, ROUGE, or perplexity – either measure surface-level similarity or the internal statistical behavior of the model, but they do not accurately reflect the quality of interaction, memory use, or conversational coherence over time. Human evaluation remains the most reliable method for assessing nuanced aspects such as contextual relevance, factual accuracy, and naturalness of dialogue, yet it is resource-intensive and difficult to scale. Therefore, more automated evaluation methods are often more suitable for extensive experimenting scenarios. In this work, BERTScore was employed as an automated metric, as it leverages contextual embeddings to better capture semantic similarity between generated and reference responses, providing a meaningful proxy for assessing the chatbot's ability to retain and convey relevant information [20]. Additionally, obtaining datasets with authentic, multi-turn conversations and annotated ground truth remains a limiting factor. Many public datasets are synthetic or domain-specific, making it hard to generalize findings.

5.1 Dataset description and preparation

To evaluate chatbot models on their ability to process and recall information from realistic conversations, a dataset REALTALK was used, including actual multi-turn human dialogues [21]. Each session (e.g., session_1) contained a series of timestamped messages with metadata including speaker name, datetime, and a unique dialogue identifier. Unlike many structured dialogue datasets, the communication in this corpus was informal and unstructured – participants responded asynchronously, with natural delays between messages and no enforced turn-taking. This format mirrors real-world messaging

applications more closely than task-oriented dialogue corpora. An example of the original data format is shown in listing 5.1.

Listing 5.1 – Dataset sample

```
{
    "clean_text": "Hi, I’m doing good how are you?",
    "speaker": "elise",
    "date_time": "30.12.2023, 00:32:20",
    "img_file": [],
    "img_url": [],
    "dia_id": "D1:2"
}
```

To use this data with LangChain and LangGraph, it was necessary to transform the structure to match the expected input format of these frameworks. These tools support only predefined roles such as «user», «ai», «assistant», or «system», and do not natively support simulating conversations between two arbitrary human speakers. As a result, all original names and role labels were anonymized and mapped to «user» and «ai», with the intention of maintaining dialogue coherence while adhering to system requirements. The reformatted structure follows the convention displayed in listing 5.2.

Listing 5.2 – Formatted dataset sample

```
{
    "role": "user",
    "content": "Hey! How are you?"
},
{
    "role": "ai",
    "content": "Hi, I’m doing good how are you?"
}
```

Each session was grouped into longer blocks of approximately 70 messages to simulate extended interaction and to populate a context window with a big chunk of information, which is essential for testing long-term memory behavior and multi-turn context tracking. This preprocessing choice balances realism with practical constraints related to model input limits and retrieval scope.

To evaluate how well the models could retain and retrieve factual content, ten factual questions were manually authored per session. These questions targeted information distributed across both the early and later parts of the session, allowing for balanced assessment of short- and long-range memory usage (listing 5.3). The manual question creation ensured that each query corresponded to a clearly identifiable fact mentioned in the session, such as a specific name, date, or decision. Also the questions contained the short answer to evaluate a BERTScore in the future. This choice was critical, as it allowed for objective comparison between generated and expected responses.

Listing 5.3 – Sample of questions

```
{
   "question": "What class was going user to take",
   "expected_answer": "Cooking class"
 },
 {
   "question": "In which city ai is going to go to bar with friends",
   "expected_answer": "Miami"
 }
```

Despite these efforts, some limitations remain. The role substitution can introduce representational bias and make certain conversational patterns less

realistic. Additionally, although the dataset reflects authentic human communication, it lacks diversity across domains and speaker demographics, which limits generalizability.

5.2 Justification for using BERTScore

Evaluating the quality of chatbot responses is inherently challenging, particularly when the goal is to measure semantic similarity and the preservation of factual information rather than exact word overlap. Traditional automatic metrics like BLEU or ROUGE primarily focus on surface-level n-gram overlap, which often fails to capture the true meaning or adequacy of a response in conversational settings. Because chatbot outputs tend to be more varied and less formulaic than machine translation or summarization outputs, relying solely on these metrics can be misleading.

To address these limitations, BERTScore was employed as a more semantically aware evaluation metric. BERTScore leverages contextual embeddings from pretrained transformer models (such as BERT) to compute similarity scores between generated responses and reference answers based on token-level semantic alignment rather than exact lexical matching. It measures three core metrics: precision, recall, and F1 score. Precision measures how much of the generated content is semantically relevant to the reference answer by comparing generated tokens to the reference. Recall reflects how well the generated output covers the expected answer by matching reference tokens to the generated ones. F1 score serves as a harmonic mean of precision and recall, offering a balanced view of both adequacy and completeness [22]. This allows it to better capture paraphrases, synonyms, and nuanced language variations that are common in natural conversations.

While BERTScore is not a perfect substitute for human evaluation, it provides a meaningful quantitative proxy that can automatically assess how closely the chatbot’s answers reflect the expected facts and intent. It is particularly

useful for detecting whether the model retains core information and answers questions accurately without requiring expensive or time-consuming human judgments for every test case. In summary, BERTScore was chosen because it offers a practical balance between automation and semantic relevance, enabling scalable evaluation of conversational quality and factual consistency in the absence of extensive human annotation.

5.4 Implementation of the BERTScore

While BERTScore provides a useful automated metric for evaluating the similarity between model outputs and expected answers, its effectiveness diminishes in cases where the model generates incorrect, irrelevant, or entirely missing responses. During experimenting, it was observed that BERTScore sometimes assigned high scores even when the model failed to answer the question correctly or at all. To address this, a custom evaluation function was implemented to penalize such outputs. In all models' prompts such phrase was added: «If the context doesn't contain relevant information, say "No answer"». Specifically, if a model's response matched a predefined non-answer, that instance was marked and manually adjusted to reflect a «[NO ANSWER]» token, while the corresponding expected answer remained unchanged. After calculating standard BERTScore values for all pairs, any penalized responses were then explicitly set to zero for precision, recall, and F1. This adjustment allowed for a more accurate and fair assessment of the model's true performance by preventing inflated scores from non-informative outputs and aligning the evaluation more closely with real-world expectations of conversational quality.

6 RESULTS OF EXPERIMENTS

6.1 Impact of temperature on model performance

To evaluate how the model's generation parameters and context size affect performance an experiment was conducted having varied the temperature parameter (0.0, 0.25, 0.5, 0.75, and 1.0), which controls the randomness of the model's output. Lower temperatures make the model more deterministic, while higher values encourage more creative but potentially less accurate responses [23]. A fixed conversation context length of approximately 2500 tokens was used. Each conversation had its own set of questions, and five iterations were performed for each temperature setting.

6.1.1 Temperature experiment: LLM model with context

Results of the experiment are presented in table 6.1 and on figure 6.1.

Table 6.1 – Results of temperature experiment LLM model with context

Temperature	Mean BERTScore (P / R / F1)	Notes
1	2	3
0.0	0.8486 / 0.8923 / 0.8702	The answers are relatively short, they contain almost no unrelated to question information
0.25	0.8490 / 0.8896 / 0.8716	The answers are relatively short, contain almost no unrelated to question information

Table 6.1 continued

1	2	3
0.5	0.8400 / 0.8964 / 0.8670	in average 2.8/10 answers were not provided
0.75	0.8360 / 0.8960 / 0.8636	The answers are more broad, contain not only answer to a asked question, but a context of the conversation
1	0.8358 / 0.8926 / 0.8630	

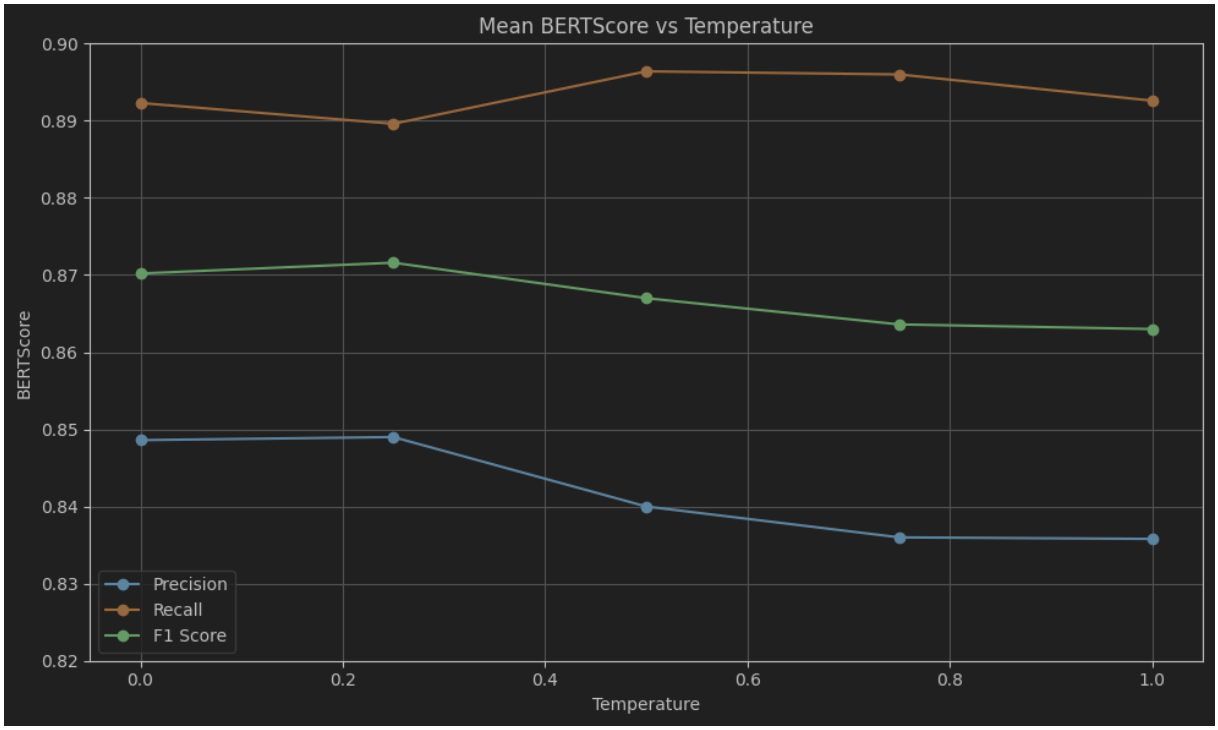

Figure 6.1 – Results of temperature experiment LLM model with context

The results indicate that lower temperatures (e.g., 0.0 and 0.25) yield slightly higher BERTScore F1 values – 0.8702 and 0.8716 respectively, suggesting more precise and focused answers. These outputs tend to be concise and contain little to no extraneous information. As the temperature increases, the

model's responses become broader and more verbose, often including contextual or additional details beyond the specific question. This shift is reflected in a slight decrease in F1 scores at temperatures 0.75 and 1.0 (0.8636 and 0.8630, respectively). The findings suggest that for tasks prioritizing factual precision and concise answering, lower temperatures are preferable when using LLMs with full conversation context.

6.1.2 Temperature experiment: LLM with summary

Results of the experiment are presented in table 6.2 and on figure 6.2.

Table 6.2 – Results of temperature experiment LLM model with context summarization

Temperature	Mean BERTScore (P / R / F1)	Notes
0.0	0.6420 / 0.6476 / 0.6444	in average 2.8/10 answers were not provided
0.25	0.6534 / 0.6528 / 0.6526	in average 2.6/10 answers were not provided
0.5	0.7106 / 0.7168 / 0.7134	in average 2.4/10 answers were not provided
0.75	0.6992 / 0.7048 / 0.7018	in average 2.6/10 answers were not provided
1	0.6946 / 0.7002 / 0.6970	in average 2.6/10 answers were not provided

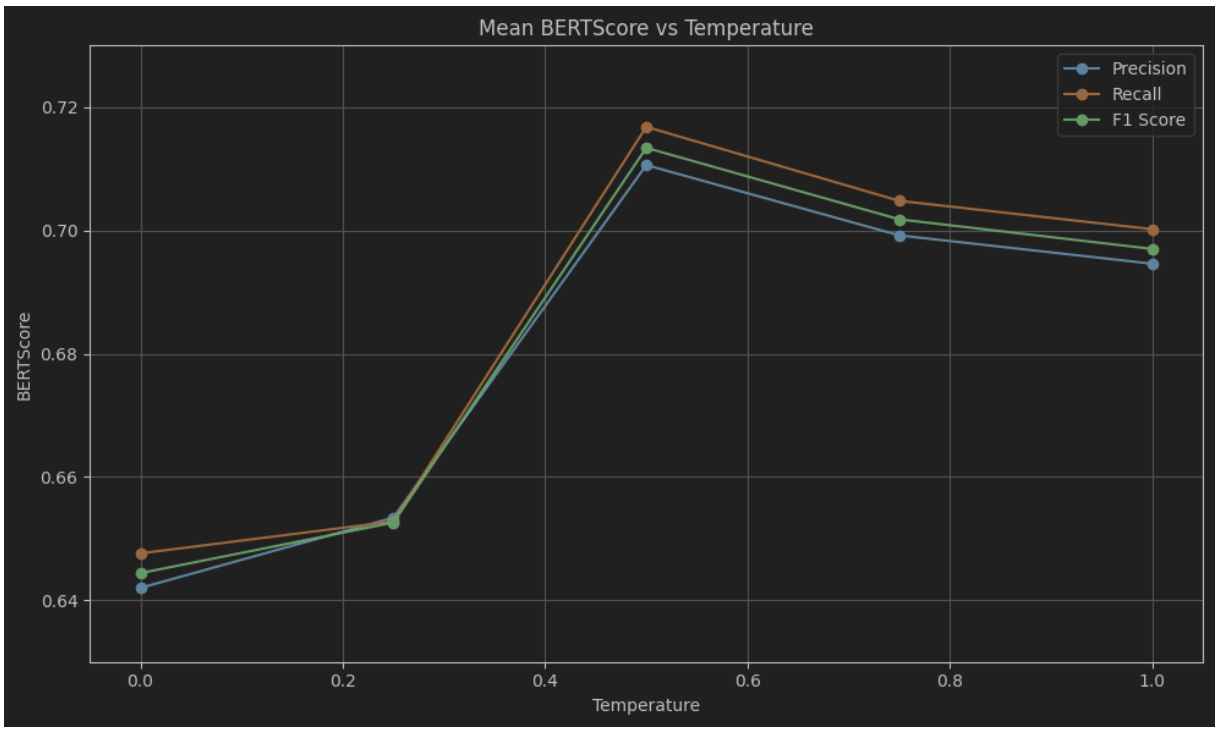

Figure 6.2 – Results of temperature experiment LLM model with context summarization

The evaluation of temperature settings in the LLM with summary configuration revealed a more pronounced sensitivity to temperature changes than in the LLM with context setup. Having set the summarization model temperature at 0.1 and at answering model's temperature 0.0, the system produced the lowest average BERTScore (P: 0.6420, R: 0.6476, F1: 0.6444), with around 2.8 out of 10 questions remaining unanswered. As temperature increased to 0.5, both performance and response rates improved, yielding the highest F1 score of 0.7134 and the fewest unanswered questions on average (2.4/10). However, further increases in temperature (0.75 and 1.0) did not maintain this upward trend. Despite more variability in responses, the model slightly declined in performance (F1: ~0.7018–0.6970) and still omitted 2.6 responses on average.

These results suggest that moderate temperature settings (especially 0.5) help the model extract and use summarized information more effectively, while higher or lower extremes either limit output diversity or introduce unnecessary

generalization or hallucination. Unlike the full-context model, the summarization-based architecture is more dependent on fine-tuned decoding behavior to balance brevity with relevance.

6.1.3 Temperature experiment: LLM with RAG

Results of the experiment are presented in table 6.3 and on figure 6.3.

Table 6.3 – Results of temperature experiment LLM model with RAG framework

Temperature	Mean BERTScore (P / R / F1)	Notes
0.0	0.7973 / 0.8197 / 0.8081	
0.25	0.7996 / 0.8192 / 0.8098	
0.5	0.8605 / 0.9004 / 0.8792	Questions may have caused deviation of the score.
0.75	0.7965 / 0.8181 / 0.8077	
1	0.7978 / 0.8253 / 0.8112	

The performance of the RAG-based system varied significantly depending on the temperature setting, with the highest BERTScore achieved at

temperature 0.5 (Precision: 0.8605, Recall: 0.9004, F1: 0.8792). For other temperature values, scores remained in the range of approximately 0.796–0.811.

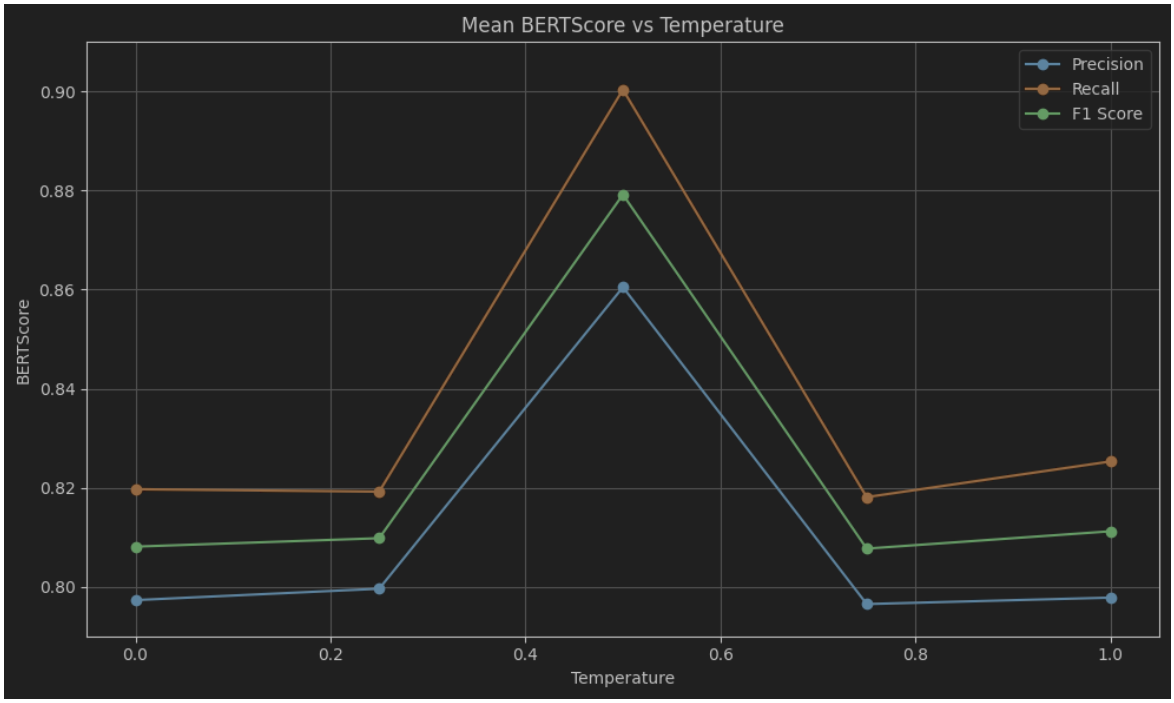

Figure 6.3 – Results of temperature experiment LLM model with RAG framework

However, during initial experimenting on the RAG system, the model consistently produced low BERTScore values of around 0.450 across all metrics, indicating a serious performance bottleneck. In an attempt to improve this, the chunk size used for document splitting was adjusted – reduced from 400 to 125 symbols with a 50-symbol overlap. Despite this change, there was no significant improvement in answer quality.

Further investigation revealed that the root cause was likely related to the quality of embeddings used in the retrieval process. The original Ollama 3.2 embeddings were replaced with OpenAI's text-embedding-3-small model, which dramatically improved retrieval relevance. As a result, the average BERTScore values increased to ~0.78 and higher in most cases, confirming that embedding quality plays a critical role in the effectiveness of RAG-based architectures. It should also be noted that the question set used during experiment may have been

slightly favorable for the model, which could have influenced the overall evaluation outcomes.

6.2 Context size experiment

The experiment was conducted with a temperature setting of 0.1, which prioritizes response precision and coherence. To evaluate how increasing memory affects performance, the input context size was varied across approximately 1000, 2500, and 5000 tokens. Each configuration was tested in five separate runs to ensure result stability. The number of questions per session was not fixed but varied depending on the context size: shorter contexts (1000 tokens) often allowed fewer than 10 questions, while longer contexts (5000 tokens) enabled more extended interactions, sometimes exceeding 10 questions.

6.2.1 Context size experiment: LLM with context

Results of the experiment are presented in table 6.4 and on figure 6.4.

Table 6.4 – Results of context size experiment LLM model with context

Context size (tokens)	Precision mean	Recall mean	F1 mean
1,000	0.834	0.864	0.849
2,500	0.833	0.862	0.847
5,000	0.849	0.886	0.867

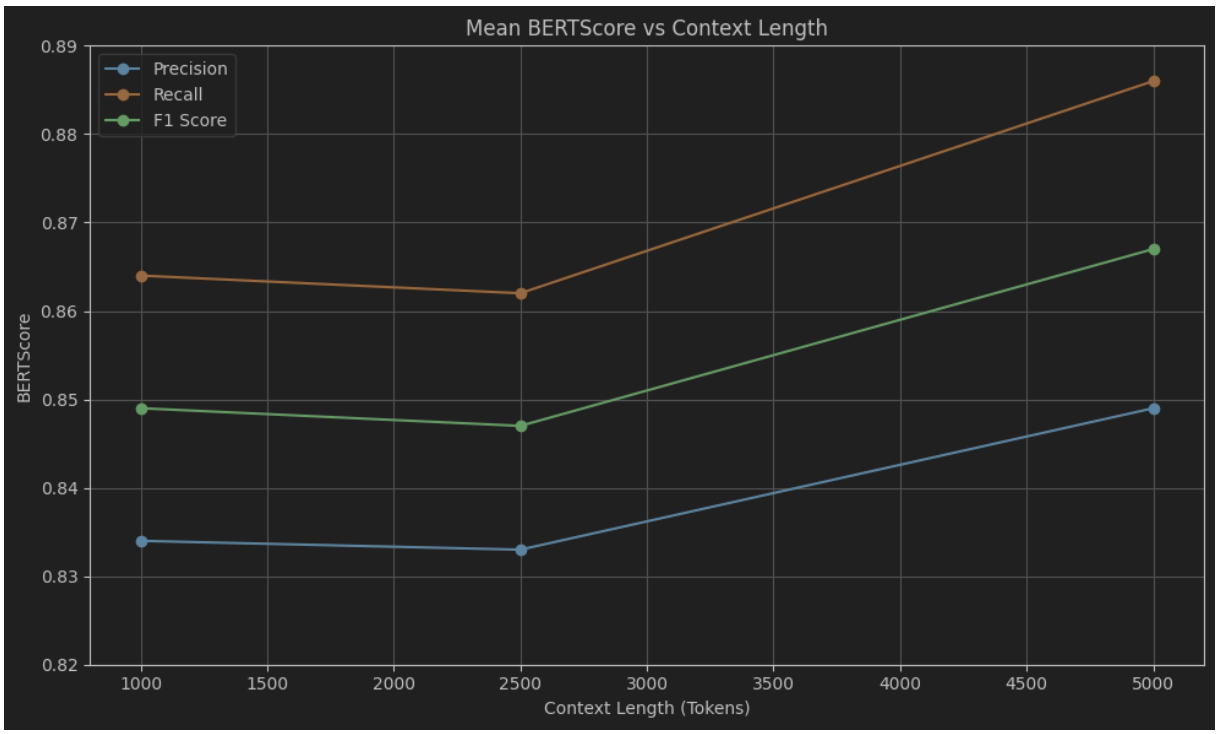

Figure 6.4 – Results of context size experiment LLM model with context

The BERTScore results demonstrated consistently high performance across all three settings, with only minor variations. Specifically, the F1 scores were 0.849, 0.847, and 0.867, respectively. These results suggest that the model performs well even with a reduced context, though a slight improvement in output quality was observed when increasing the context to 5000 tokens. This indicates that while the LLM is capable of leveraging shorter contexts effectively, providing it with more conversational history can marginally enhance its ability to produce factually accurate and contextually rich answers.

6.2.2 Context size experiment: LLM with summary

Results of the experiment are presented in table 6.5 and on figure 6.5.

Table 6.5 – Results of context size experiment LLM model with context summarization

Context size (tokens)	Precision mean	Recall mean	F1 mean
1,000	0.657	0.660	0.658
2,500	0.675	0.671	0.673
5,000	0.689	0.685	0.687

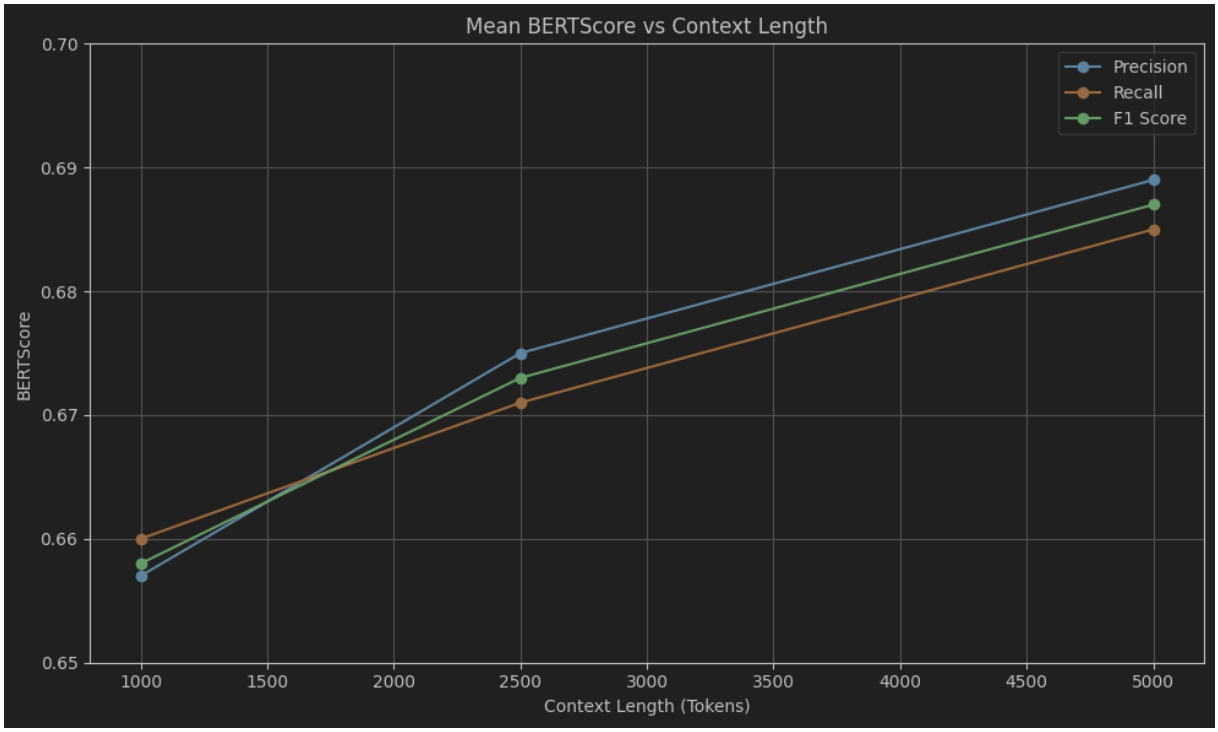

Figure 6.5 – Results of context size experiment LLM model with context summarization

The results showed a gradual increase in BERTScore F1 from 0.658 (1k tokens) to 0.673 (2.5k tokens), and finally to 0.687 (5k tokens). While this indicates that larger contexts may lead to slightly more useful summaries, the performance remained significantly lower compared to using the full conversation context. This suggests that, although summarization can be an efficient memory-saving approach, it tends to sacrifice some factual accuracy and detail necessary for high-quality responses – particularly in complex or multi-turn dialogues.

6.2.3 Context size experiment: LLM with RAG framework

Results of the experiment are presented in table 6.6 and on figure 6.6.

Table 6.6 – Results of context size experiment LLM model with RAG framework

Context size (tokens)	Precision mean	Recall mean	F1 mean
1,000	0.761	0.792	0.776
2,500	0.761	0.792	0.776
5,000	0.739	0.762	0.750

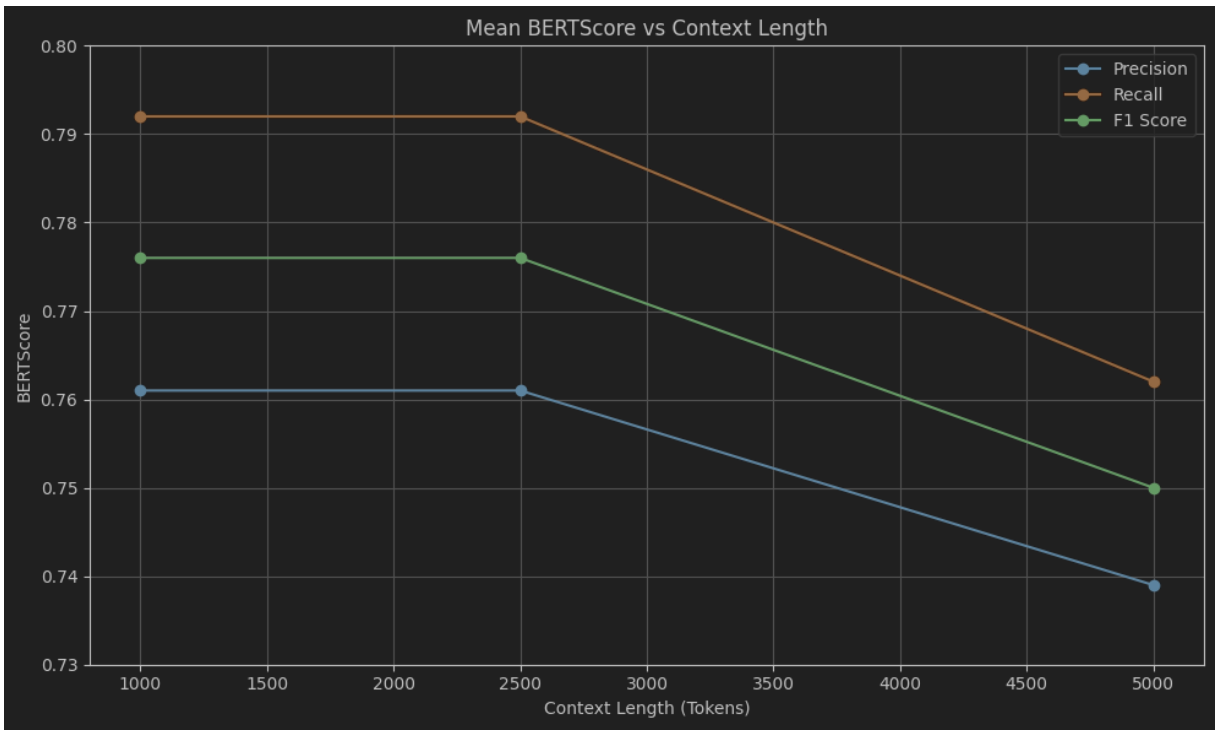

Figure 6.6 – Results of context size experiment LLM model with RAG framework

The F1 scores remained constant at 0.776 for both 1k and 2.5k token contexts, while a slight drop to 0.75 was observed at 5k tokens. These results suggest that simply expanding the context size does not guarantee better performance in retrieval-augmented generation. One likely explanation is that larger input contexts may introduce noise or dilute relevant information, making retrieval less effective.

The retrieval process relies heavily on the quality of the document chunking strategy and the embedding model used to represent those chunks in a vector space. Poor chunking granularity or suboptimal embeddings can significantly degrade the system's ability to locate contextually relevant passages, regardless of input length. The drop in F1 score at 5k tokens may therefore be attributed to an increased presence of semantically weak or distracting content within the extended context window, leading to less focused retrieval.

6.3 Large context experiment

The third test was a limit experiment, where 18k token length conversation was passed to the model with temperature 0.1 and 10 questions randomly with the same seed for all models were asked.

Results of the experiment are presented in table 6.7 and on figure 6.7.

Table 6.7 – Results of high context size experiment of 3 LLM configurations

Model	Context size (tokens)	Precision mean	Recall mean	F1 mean
LLM with context	18,000	0.840	0.898	0.868
LLM with summary	18,000	0.709	0.706	0.707
LLM with RAG framework	18,000	0.883	0.901	0.892

The large context experiment was conducted using an extended input length of approximately 18,000 tokens to explore how scaling context size affects the performance of different chatbot architectures. The LLM with context model maintained strong results, with an F1 score of 0.868, demonstrating its ability to handle very large inputs effectively. The LLM with summary approach showed moderate improvement compared to shorter contexts, achieving an F1 of 0.707, which suggests that summarization benefits from increased context but still trails behind direct context usage.

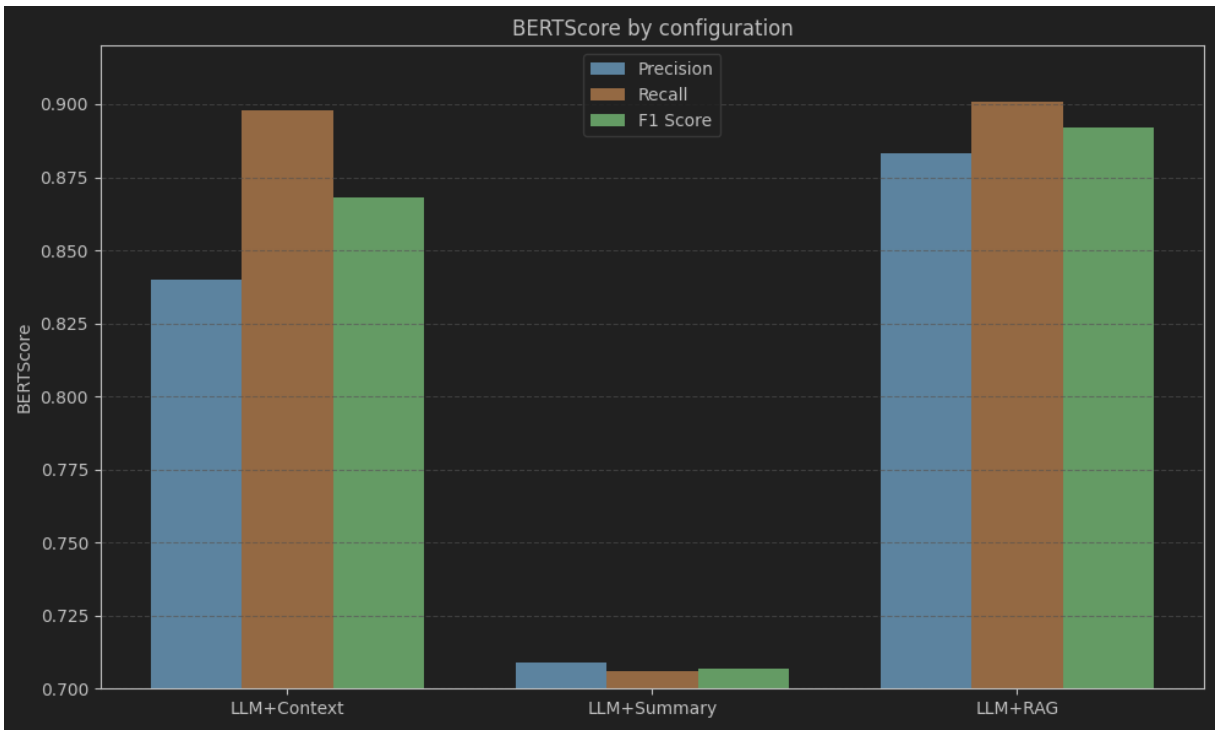

Figure 6.7 – Results of high context size experiment of 3 LLM configurations

Notably, the LLM with RAG framework system achieved the highest performance, with an F1 score of 0.892, surpassing the other two methods. This indicates that retrieval-augmented generation can leverage large document collections more effectively than simple summarization or direct context input, likely due to more precise access to relevant information spread across vast content.

7 FRAMEWORK OF AI CHATBOT SELECTION

The goal of this research is to propose a framework for selecting a chatbot based on the needs of users. Through systematic evaluation, three distinct configurations of LLMs were tested under different context lengths, temperature settings, and the experiment results will form the basis of the framework..

The LLM with full context consistently demonstrated the most stable and high-performing results across all tests. The F1 score remained high in cases with short (0.849), medium (0.847), and long (0.867) contexts, and it achieved an F1 score of 0.868 in an 18k-token test.

In contrast, the LLM with summary showed the lowest performance across all tests. Despite this, it showed a clear trend of improvement with increased context size, rising from an F1 score of 0.658 at 1k tokens to 0.687 at 5k, and achieving 0.707 in the 18k-token test.

The LLM with RAG performed poorly when suboptimal embedding models and chunking strategies were used. Early results showed low F1 scores around 0.431, but after replacing the embedding model from Ollama3.2 to text-embedding-3-small and adjusting chunk sizes, performance substantially improved, achieving a mean F1 of 0.776 for 1k and 2.5k token contexts and reaching the highest observed score (0.892) at 18k tokens.

7.1 Framework logic

For the chatbot decision framework (figure 7.1), the context length plays the most important role and serves as a starting point. As this parameter directly influences both the precision of the responses, because it determines how relevant the retrieved information is, and the accuracy of the overall output, ensuring that the chatbot's decisions are based on coherent and sufficient input.

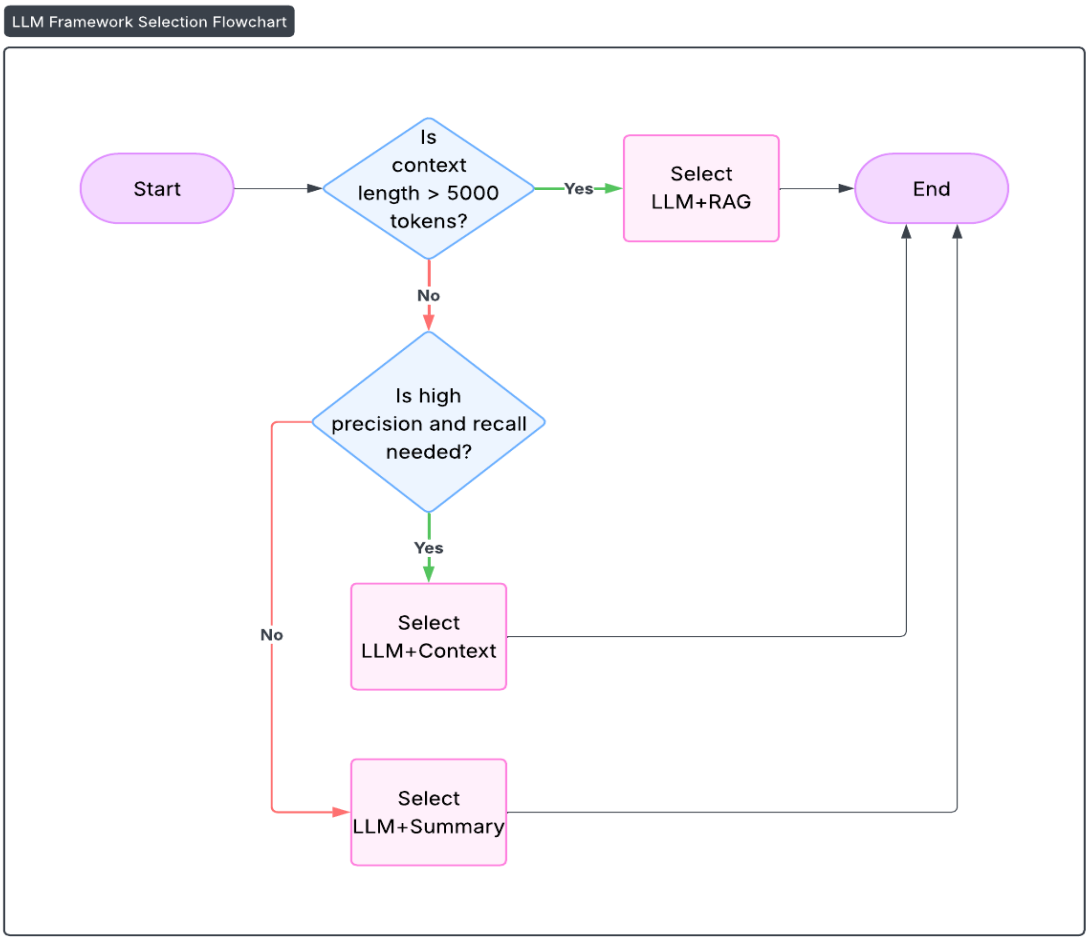

Figure 7.1 – LLM configuration selection framework

The LLM selection process should follow this flow:

a) if the context history or accompanying documents exceed 5,000 tokens, the recommended configuration is LLM with RAG. This architecture demonstrates the highest performance in large context cases, achieving an F1 score of 0.892 in an 18k-token scenario. Such results prove its suitability for multi-session, document-rich conversations;

b) if the context history is less than 5,000 tokens, the selection should be based on the level of precision and recall:

– if high accuracy and precision are required – LLM with context is the optimal choice. It consistently performed well across different context lengths with an F1 score up to 0.867. As for the model provider, high

context window models should be used, especially the latest Google Gemini models (Gemini-1.5-Pro, Gemini-2.0-Flash) or OpenAI’s GPT-4.1 with a 1M context window;

– if high accuracy is not essential and more generalized information is needed or memory usage is limited – LLM with summary is a viable choice.

As for the temperature selection, as a rule of thumb, a temperature that is between 0.25 and 0.5 should be selected. It keeps the models from hallucinating and introducing off-topic elaborations, as high temperatures did not show improvements to the models’ performance.

CONCLUSIONS

This research investigated the performance of various chatbot configurations based on LLMs and tested their suitability for emulating human-like interactions across different context sizes. Three architectures were tested: LLM with direct context, LLM with summary, and LLM with RAG framework. Performance was measured using the BERTScore across precision, recall, and F1 score, with experimental temperature and context length as experimental variables.

The results showed that the LLM with context is the most well-rounded and stable configuration, maintaining high performance among short and very large contexts, with an F1 score of up to 0.867.

LLM with RAG serves as an optimal solution for handling long or document-intensive contexts, especially at 18k context, where it achieved the highest F1 score (0.892) across all configurations, demonstrating its applicability in multi-session or high-volume data retrieval tasks.

In contrast, the LLM with summary showed the lowest performance across all the tests, but had small improvements with larger context sizes, making it suitable in memory-constrained scenarios where high precision and recall is not required.

An important outcome of the research is the development of a decision-making framework that guides the selection of the appropriate chatbot based on context length, accuracy requirements, and temperature settings. Its relevance is particularly high in fields such as digital assistants, customer support, or interactive knowledge management.

The decision-making framework recommends:

– LLM + RAG if the context length exceeds 5,000 tokens;

– LLM + Context for shorter contexts requiring high precision and recall;

– LLM + Summary for efficiency-based applications with low performance needs.

This study opens several directions for further research. One direction is the exploration of adaptive hybrid models that dynamically switch between configurations during runtime based on user behavior and query complexity. Another – deeper investigation of memory optimization mechanisms that may help to improve long-term usability and realism of chatbots, providing a better user experience.

In summary, this thesis contributes to the foundational understanding of LLM-based chatbot performance and offers a scalable, test-driven framework that supports the design of efficient, responsive, and contextually aware conversational agents.

REFERENCES

1. Dictionary.com | meanings & definitions of english words. *Dictionary.com*. URL: https://www.dictionary.com/browse/chatbot (date of access: 29.05.2025).

2. Turing A. M. I.–computing machinery and intelligence. *Mind*. 1950. Vol. LIX, no. 236. P. 433–460. URL: https://doi.org/10.1093/mind/lix.236.433 (date of access: 29.05.2025).

3. Weizenbaum J. ELIZA – a computer program for the study of natural language communication between man and machine. Communications of the ACM. 1966. Vol. 9, no. 1. P. 36 – 45. URL: https://doi.org/10.1145/365153.365168 (date of access: 02.04.2025).

4. Attention is all you need. *arXiv.org*. URL: https://arxiv.org/abs/1706.03762 (date of access: 29.05.2025).

5. Church B., Finn T. Types of chatbots | IBM. *IBM – United States*. URL: https://www.ibm.com/think/topics/chatbot-types (date of access: 29.05.2025).

6. BERT: pre-training of deep bidirectional transformers for language understanding. *arXiv.org*.
URL: https://doi.org/10.48550/arXiv.1810.04805 (date of access: 29.05.2025).

7. Language models are few-shot learners. *arXiv.org*. URL: https://arxiv.org/abs/2005.14165 (date of access: 29.05.2025).

8. An image is worth 16x16 words: transformers for image recognition at scale. *arXiv.org*. URL: https://arxiv.org/abs/2010.11929 (date of access: 29.05.2025).

9. Stryker C. What is AI agent memory? | IBM. *IBM – United States*. URL: https://www.ibm.com/think/topics/ai-agent-memory (date of access: 29.05.2025).

10. Martineau K. What is retrieval-augmented generation (RAG)?. *IBM Research*. URL: https://research.ibm.com/blog/retrieval-augmented-generation-RAG (date of access: 29.05.2025).

11. Think topics | IBM. *IBM – United States*. URL: https://www.ibm.com/think/topics/rag-techniques. (date of access: 29.05.2025).

12. Barnard J. What is embedding? | IBM. *IBM – United States*. URL: https://www.ibm.com/think/topics/embedding (date of access: 29.05.2025).

13. What is similarity search? | pinecone. *The vector database to build knowledgeable AI | Pinecone*. URL: https://www.pinecone.io/learn/what-is-similarity-search/ (date of access: 29.05.2025).

14. Gemini 2.0 flash | generative AI on vertex AI | google cloud. *Google Cloud*. URL: https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-0-flash (date of access: 29.05.2025).

15. Introducing GPT-4.1 in the API | *OpenAI*. URL: https://openai.com/index/gpt-4-1/ (date of access: 21.05.2025).

16. Introducing claude 3.5 sonnet. *Home \ Anthropic*. URL: https://www.anthropic.com/news/claude-3-5-sonnet (date of access: 29.05.2025).

17. Large enough | mistral AI. *Frontier AI LLMs, assistants, agents, services | Mistral AI*. URL: https://mistral.ai/news/mistral-large-2407 (date of access: 29.05.2025).

18. Mixtral of experts | Mistral AI. *Frontier AI LLMs, assistants, agents, services | Mistral AI*. URL: https://mistral.ai/news/mixtral-of-experts (date of access: 29.05.2025).

19. Introducing Llama 3.1: our most capable models to date. *AI at Meta*. URL: https://ai.meta.com/blog/meta-llama-3-1/ (date of access: 29.05.2025).

20. GitHub - Tiiiger/bert_score: BERT score for text generation. *GitHub*. URL: https://github.com/Tiiiger/bert_score (date of access: 29.05.2025).

21. GitHub – danny911kr/REALTALK: AI agent memory evaluation benchmark based on long-term, real-world conversational data (not LLM-simulated dialogues). *GitHub*. URL: https://github.com/danny911kr/REALTALK (date of access: 29.05.2025).

22. Classification: accuracy, recall, precision, and related metrics | machine learning | google for developers. *Google for Developers*. URL: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall (date of access: 29.05.2025).

23. D J. M. P., Noble J. What is LLM temperature? | IBM. *IBM – United States*. URL: https://www.ibm.com/think/topics/llm-temperature (date of access: 29.05.2025).