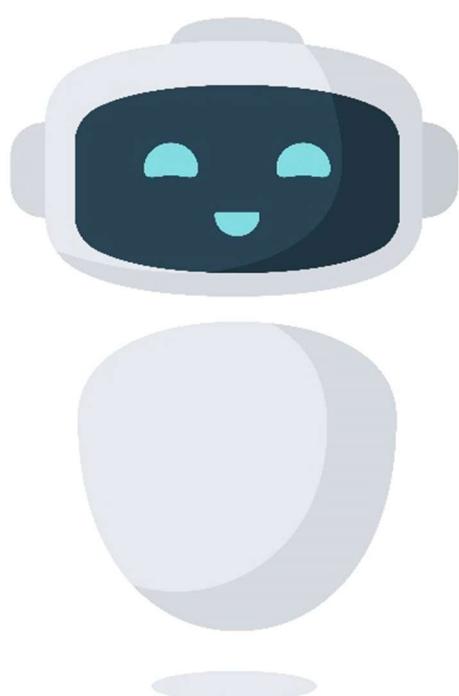


// NAVATAR-Helper

*Use of Retrieval Augmented Generation (RAG) Framework in
The Development of an LLM-based Healthcare-Chatbot*



// Team

- Sirin Koca – s182211
- Younes Benhaida - s374220
- Rafey-Ul-Islam Afzal - s374150
- Morten Nilsen - s371473
- Valerie Smidt - s374206

// Supervisor: Associated Professor Hårek Haugerud

// Course: DATA3750 – Applied Artificial Intelligence (AI) & Data Science Project

// Institution: Oslo Metropolitan University, Department of Computer Science, H24

Table of Contents

Table of Contents	1
Abstract.....	3
Acknowledgements.....	3
Executive Summary.....	4
Glossary	5
1. INTRODUCTION	6
2. BACKGROUND.....	6
2.1 Context – The Problem	6
2.2 Stakeholders.....	7
2.3 Related Work	7
3. METHODOLOGY.....	8
3.1. RAG Framework	8
3.1.1 Types of RAG.....	8
3.2 Large Language Models (LLM).....	9
3.3 Data Preprocessing	9
4. PROGRESS: process & product.....	10
4.1. PROCESS.....	10
4.1.1 An Important Aspect: Hallucinations	10
4.1.2 Evaluation Strategy - Minimizing Hallucinations	11
4.1.3 Hallucination Rate	11
4.1.4 Metrics Used	11
4.2. PRODUCT: NAVATAR-Helper.....	12
4.2.1 Main Components.....	12
4.2.2 The LLM - NorMistral-7b.....	13
4.2.3 The VectorDB - Milvus Lite	13
5. SYSTEM ARCHITECTURE & Data Pipelines	15
6. TESTING.....	18
6.1 Manual Testing:	18
6.1.1 Benchmarking Against Non-RAG Baseline.....	18

6.1.2 The 100 Question Experimentation:	19
6.2 Automated Testing:	22
6.3 User Testing:	24
7. RESULTS	25
8. DISCUSSIONS	27
8.1 Technical Challenges	27
8.2 Lessons Learned	27
9. FUTURE WORK	28
10. CONCLUSION	29
11. APPENDIX	30
11.1 Backend – RAG Framework	30
11.2 Vector Embeddings and Vector Database	31
11.3 GPU Server.....	32
11.4 Frontend – Streamlit UI	34
11.5 Testing Process	34
11.6 The Final Tech-Stack.....	36
11.7. System Flow.....	36
11.8 User Manual.....	37
11.9 User Survey.....	38
11.10 User Survey – Results	39
11.11 Collaboration & Documentation	41
11.12 Project Plan & Management.....	41
11.13 GitHub Repository & Code Base.....	42
11.14 Project Midterm Presentation	43
11.15 Future of AI.....	44
12. REFERENCES	45
Figure List	47

Abstract

The **NAVATAR-Helper** project-team at OsloMet developed a **bilingual AI chatbot** leveraging the **Retrieval-Augmented Generation (RAG)** framework to address **NEET (Not in Education, Employment, or Training)**-related health queries in Norwegian and English. The system prioritized accuracy and minimized hallucinations by retrieving verified information from a curated NEET document dataset using advanced vector embeddings, the Milvus database, and the NorMistral LLM.

Launched on August 30, 2024, under Associate Professor Hårek Haugerud's guidance, the project demonstrated high accuracy, reduced hallucination rates, real-time responses, and reliable source referencing through rigorous manual and automated evaluations. **NAVATAR-Helper** exemplifies responsible AI application in healthcare NLP, establishing a scalable foundation for future public sector use.

Acknowledgements

We would like to show our gratitude and thank everyone who supported and guided us along the way. Special thanks to our supervisor, Associate Professor Hårek Haugerud, for his dedication and for presenting us with such a challenging and inspiring project.

Also, thanks to our expert panel for their insights during the user-testing phase, and the employees at the OsloMet AI-Lab for providing a working space with a GPU-server dedicated to this project. This report is the result of effective teamwork, clear communication, and collective collaboration, which have been vital to our success.

Executive Summary

NAVATAR-Helper advances domain-specific AI with a RAG-framework, delivering accurate, bilingual responses to NEET-specific health inquiries. The project's key achievement is a **25% reduction in hallucinations** through verified NEET data, confirmed by rigorous testing. Built on OsloMet's GPU infrastructure, the chatbot provides reliable, real-time responses, supporting healthcare professionals and social service providers.

Key Metrics:

- **Accuracy Rate:** Achieved over 85% accuracy in providing relevant answers during controlled testing sessions - chatbot's performance is measured using predefined, carefully monitored test scenarios.
- **Hallucination Reduction:** Reduced hallucination rates by **25% compared to a non-RAG baseline**, ensuring that responses were grounded in validated sources.
- **Response Time:** The system maintained an average response time of **under 30 seconds**, meeting user expectations for real-time interaction.
- **User Satisfaction:** Feedback from expert panel tests rated the **chatbot's reliability at 4.5 out of 5** on average.

Glossary

TERM	DEFINITION
Artificial Intelligence (AI)	A field of study focused on creating systems that simulate human intelligence through computational methods.
Base Model	The initial version of a large language model (LLM) trained on general data, requiring fine-tuning for specific tasks (e.g., Mistral-7b-v0.1).
Chunk	A segment of text, such as a paragraph or sentence, used in processing to maintain context and coherence.
Context	The portion of input text, measured in tokens, that an LLM considers when generating a response.
Ground Truth	Verified and factual data used as a standard for training and evaluating AI models.
Hallucination	Instances where an LLM produces information not found in the input data, resulting in potentially misleading outputs.
LangChain	A framework that connects LLMs with external tools and data sources, enabling complex AI applications like RAG.
Large Language Model (LLM)	An AI model trained on vast amounts of text data to generate human-like text responses.
Prompt Engineering	The process of crafting prompts to steer LLM outputs towards desired responses.
Retrieval-Augmented Generation (RAG)	An AI framework that combines information retrieval from a database with LLM-generated content to enhance response accuracy.
Scalability	The capacity of the system to handle growth, adapting to larger datasets and more extensive use cases.
Similarity Search	A technique that finds items in a database most similar to a given query, often using vector embeddings.
Token	An individual unit of text (e.g., a word or symbol) processed by an LLM.
Tokenization	The process of breaking down text into individual tokens, such as words or sub-words.
Vector Database	A database designed to store and retrieve vector embeddings efficiently for similarity searches.
Vector Embedding	The numerical representation of text or words in a multi-dimensional vector space, facilitating similarity comparisons.

Figure 1- Terminology

1. INTRODUCTION

The **NAVATAR-Helper** project addresses a critical need in healthcare communication by delivering reliable, NEET-specific information to counsellors and researchers. Built on the RAG framework, this bilingual chatbot ensures accurate, hallucination-free responses in Norwegian and English by utilizing curated NEET documents, advanced language models, embedding techniques, and vector databases for robust performance.

2. BACKGROUND

Our collective interest in AI and machine learning brought us together for this project. Several team members, including Sirin, had prior experience in AI, having completed projects such as a bachelor's thesis in machine learning in collaboration with SINTEF. This foundational interest guided our decision to pursue DATA3750 – Applied AI and Data Science Project at OsloMet.

The NAVATAR-Helper project was presented by Associate Professor Hårek Haugerud, who invited us to join an initial meeting after expressing our interest. At this meeting, we connected as a five-member team, each passionate about AI and eager to take on the project. Professor Haugerud provided an overview of the project's goals, particularly emphasizing the integration of a RAG-framework and the critical challenge of addressing and reducing hallucinations—instances where AI produces unsupported or fabricated information. This focus resonated with us, shaping our project strategy and objectives from the start.

2.1 Context – The Problem

This project is part of a broader effort to leverage AI to address the informational needs of professionals working with NEET (Not in Education, Employment, or Training) individuals. NEET- professionals often lack accessible, structured guidance on health-related issues, and NAVATAR-Helper aims to fill this gap by creating a bilingual and reliable chatbot that can answer queries in both Norwegian and English.

2.2 Stakeholders

Key stakeholders may include healthcare professionals, NEET organizations, OsloMet researchers, domain-specific individuals, and public health agencies. The chatbot leverages cutting-edge AI technology to provide reliable, bilingual responses tailored to diverse user needs.

2.3 Related Work

A tutorial on how to build your own RAG Chatbot (Thaker, 2023) was recommended by our project supervisor as starting point. The article provided us with a baseline knowledge of RAG in order to start fast and structure our project efficiently. Additionally, an article published by the Ruter developer team (Strand 2024) served as further inspiration for our Norwegian chatbot, leading us to test various tools, databases and machine learning models such as “**intfloat/multilingual-e5-large**” and the Chroma DB as potential components for our chatbot.

3. METHODOLOGY

We used a RAG-approach to ensure responses are grounded in data, combining vector embeddings and LLMs. By combining a vector database (Milvus) with an embedding model, we optimized our system for accurate retrieval. RAG architecture facilitated precise responses while minimizing hallucinations by restricting answers to information from the NEET dataset. We selected NorMistral as our LLM due to its bilingual capabilities and tested configurations extensively to meet performance and accuracy standards (ref. section 6, Testing).

3.1. RAG Framework

RAG enhances the precision and context of responses by combining document retrieval with response generation, which helps ensure that answers are grounded in verified information. It addresses the limitations of standalone language models by fetching relevant external information to enhance response accuracy and reduce hallucinations.

3.1.1 Types of RAG

1. **Naive RAG (Native RAG):** The simplest form, involving a direct pipeline where user queries are processed through retrieval, reranking, and response generation. This setup is effective for straightforward applications but may struggle with multi-turn conversations and complex reasoning due to its lack of internal context memory.
2. **Hierarchical RAG:** An advanced version where different levels of retrieval are performed, allowing for responses that synthesize information from multiple document segments.
3. **Agentic RAG:** A more complex architecture where autonomous agents manage specific documents, with a meta-agent orchestrating their outputs. This allows for deeper analysis, multi-step reasoning, and follow-up queries, enhancing adaptability and reducing error rates.

In our project, we implemented a **Naive RAG system**. This basic RAG model suited our project's scope, enabling reliable retrieval of NEET-related data from documents and generation of accurate, single-response answers. While effective for our MVP, it highlighted limitations such as restricted multi-turn dialogue handling, a well-known constraint of native RAG setups (Zhao et al., 2024).

The implementation of RAG systems, even in basic configurations, present considerable complexities. While our system adheres to a straightforward RAG setup, the nature of its task—retrieving relevant documents and using an LLM to generate comprehensive responses—demands a multi-layered approach. As supported by Zhao et al. (2024) in their comprehensive survey on data-augmented LLMs, these systems face notable challenges, including correctly interpreting user queries, managing complex reasoning, and synthesizing data from external sources effectively. The academic paper authored by Zhao et al. (2024) reviews and categorizes the challenges, techniques, and levels of complexity involved in data-augmented LLM applications. It serves as a comprehensive study outlining the difficulties and solutions when deploying RAG systems across high-risk fields like healthcare.

Our work acknowledges these challenges and aligns with the argument that there is no universal solution for data-augmented LLM applications, as different levels of query complexity require tailored methods. This insight reinforces the complexity inherent in our RAG model, where the system must not only retrieve data but also use an LLM to bridge gaps between disparate pieces of information, ensuring the response is relevant, accurate, and reflective of professional standards (Zhao et al., 2024).

3.2 Large Language Models (LLM)

We chose NorMistral for its strong multilingual capabilities, particularly in Norwegian, ensuring precise and relevant responses. It is trained on both English and extensively Norwegian datasets made it ideal for our bilingual application.

3.3 Data Preprocessing

In our project, the domain was NEET research, with data collection from scholarly articles. We utilized a total of 130 PDFs, all of which were referenced in a scoping review (Rahmani & Groot, 2023) recommended by our supervisor. This collection of research papers provided a rich and focused dataset, offering in-depth insights into the various risk factors associated with NEET status among youth. The fact that all the articles were cited in a single scoping review ensured the relevance and quality of our dataset, making it a reliable foundation for NAVATAR-helper.

4. PROGRESS: process & product

4.1. PROCESS

Our team consists of five ambitious individuals with passion for machine learning, data science and data-driven applications. We were all excited when we met for the first time at OsloMet's AI-Lab office, located at Stensberg Gata 29, on August 30th. During our initial meeting with our supervisor, Associate Professor Hårek Haugerud, we received valuable insights about the project. Following the discussion, we quickly agreed to proceed with the planning and development phase.

Our workflow focused on two main areas: building the data pipeline (the system logic), designing main-components and fine-tuning NAVATAR-Helper's functionality. We spent the first few weeks conducting research, as none of us had prior experience in building a RAG-based chatbot. The team split the work into embedding data, configuring retrieval, and integrating the model into LangChain's framework. We conducted experiments with various configurations, including different embedding models and vector databases, to identify the optimal setup for our system (ref. section 6, Testing). NAVATAR-Helper is currently capable of responding accurately to NEET-specific queries with minimal latency. Key improvements in retrieval speed and response quality were achieved through iterative testing and optimization.

We had regular team meetings on Mondays, daily check-ins on Discord and used OneDrive and GitHub for documentation and collaboration. We also had weekly meetings with our supervisor and discussions were particularly helpful for debugging complex errors, like retrieval inconsistencies.

4.1.1 An Important Aspect: Hallucinations

In the context of AI, *hallucinations* refer to instances where LLMs generate incorrect, fabricated, or irrelevant responses that are not grounded in the available data. This is especially critical in our project, which deals with health-related queries. Any hallucinated or inaccurate response could lead to significant consequences. Thus, minimizing hallucinations is a core objective of our chatbot development. We aim to ensure that all responses are based solely on credible sources, specifically the NEET-related documents.

4.1.2 Evaluation Strategy - Minimizing Hallucinations

To assess the effectiveness of our RAG system in reducing hallucinations, we implemented a structured evaluation framework consisting of multiple metrics:

4.1.3 Hallucination Rate

The hallucination rate is defined as the percentage of responses that contain information not found in the retrieved documents. Through testing, we established a baseline threshold of 12% (ref. section 6, Testing). Any response that exceeds this threshold is flagged for review and considered a hallucination.

4.1.4 Metrics Used

4.1.4.1 Faithfulness and Accuracy

Faithfulness reflects the accuracy of the chatbot's responses based on the NEET documents. By comparing generated answers with retrieved source content, we ensure factual alignment. To achieve consistent and deterministic results, we set the language model's temperature to 0.1, minimizing creative deviations and grounding responses strictly in retrieved data.

Prompt engineering further directed the model to depend exclusively on vector database content, enhancing reliability. While this configuration minimized hallucinations, temperature is not a post-generation metric for evaluating response faithfulness. It is a pre-generation control for output randomness. To assess response reliability, we conducted systematic testing, including manual and automated evaluations, confirming alignment with the NEET dataset. This approach ensured consistent, accurate, and sourced outputs from the chatbot.

4.1.4.2 Contextual Relevance & Precision

Context recall measures how much of the relevant information is retrieved from the vector database (Milvus), while context precision evaluates how much of the retrieved information is relevant to the query. In our tests, we used the similarity distance calculations to measure context precision. Our results indicated an 88 % precision, ensuring that most retrieved documents were relevant to the query. Now, we will discuss and take a closer look at the product - NAVATAR-Helper.

4.2. PRODUCT: NAVATAR-Helper

NAVATAR-Helper is our end product - an AI chatbot built with a robust RAG-system that provides accurate answers to NEET-related queries in Norwegian and English. The chatbot's interface is user-friendly, enhancing interaction and trust by including source citations with responses.

4.2.1 Main Components

COMPONENT	DESCRIPTION
LLM	NorMistral-7b-warm-instruct chosen for bilingual capabilities, trained on Norwegian and English data for accurate, contextually relevant responses.
Embedding Model	Alibaba-NLP/gte-multilingual-base used to convert text into vector embeddings for semantic understanding and retrieval.
Vector Database	Milvus Lite chosen for scalability and efficient retrieval, storing vectorized embeddings from NEET documents.
Framework & Orchestration	LangChain coordinates the LLM and vector database, ensuring seamless integration between retrieval and generation processes.
Data Ingestion & Preprocessing	A process of loading, chunking, and embedding NEET-specific PDFs into the vector database using PyPDFDirectoryLoader for retrieving PDFs and RecursiveCharacterTextSplitter for chunking. RecursiveCharacterTextSplitter splits text using line or paragraph breaks as separators. Both libraries are from LangChain.
Frontend User Interface	Python-based Streamlit web application for users to input questions and receive answers with source citations.
Evaluation & Testing Suite	Manual, automated, and user testing processes to assess accuracy, response relevancy, and hallucination rates.

Figure 2 - Main Components of NAVATAR-Helper

The source code of the NAVATAR-Helper project is available on the GitHub repository: <https://github.com/sirin-koca/NAVATAR-Helper>

4.2.2 The LLM - NorMistral-7b

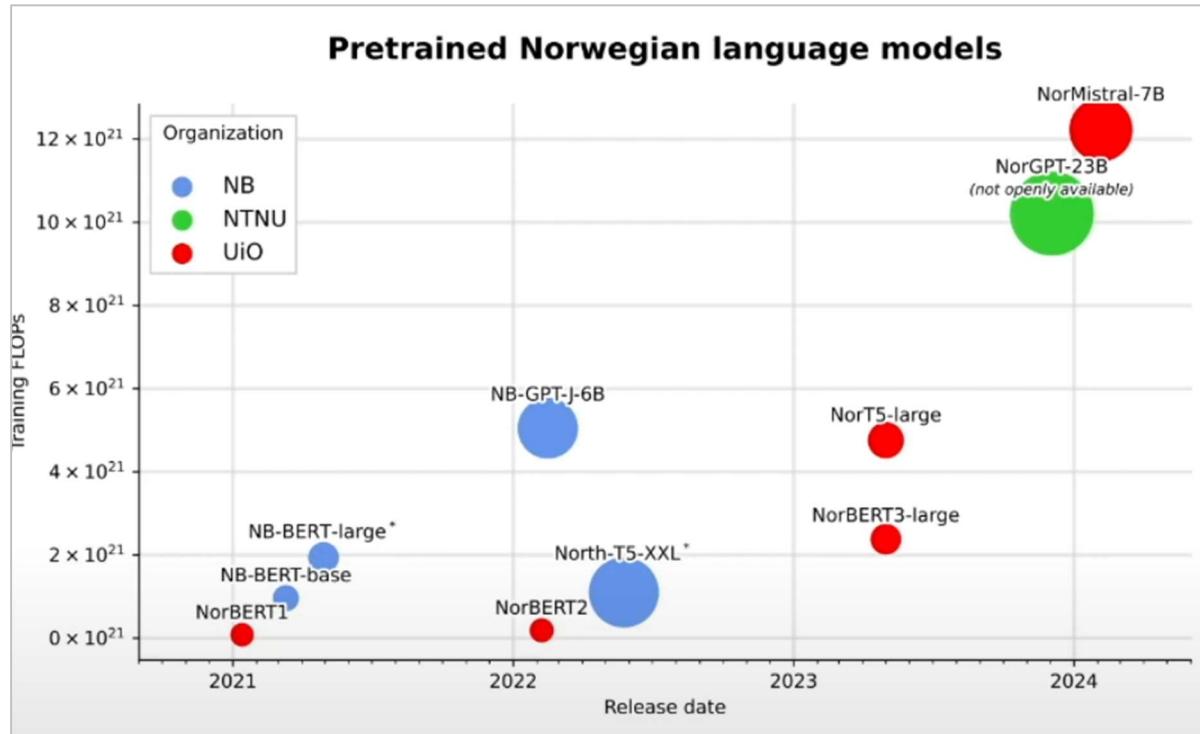


Figure 3- Publicly available local LMs (ref. section 12, References)

Figure 3 represents the Training FLOPs for various Norwegian Language Learning Models (LLMs), providing an indication of their computational complexity. The model with the highest number of Training FLOPs, positioned at the top, is NorMistral-7b. This indicates that NorMistral-7b is the most computationally intensive model available today, having undergone the most floating-point operations during its training phase.

4.2.3 The VectorDB - Milvus Lite

During the research phase of our project, we tested various tools and vector databases. One of these tests, conducted in early September, occurred when we had not yet finalized which dataset and documents to be used. As a result, we posed a general query to assess and demonstrate the performance of Milvus compared to FAISS, specifically evaluating its speed advantage.

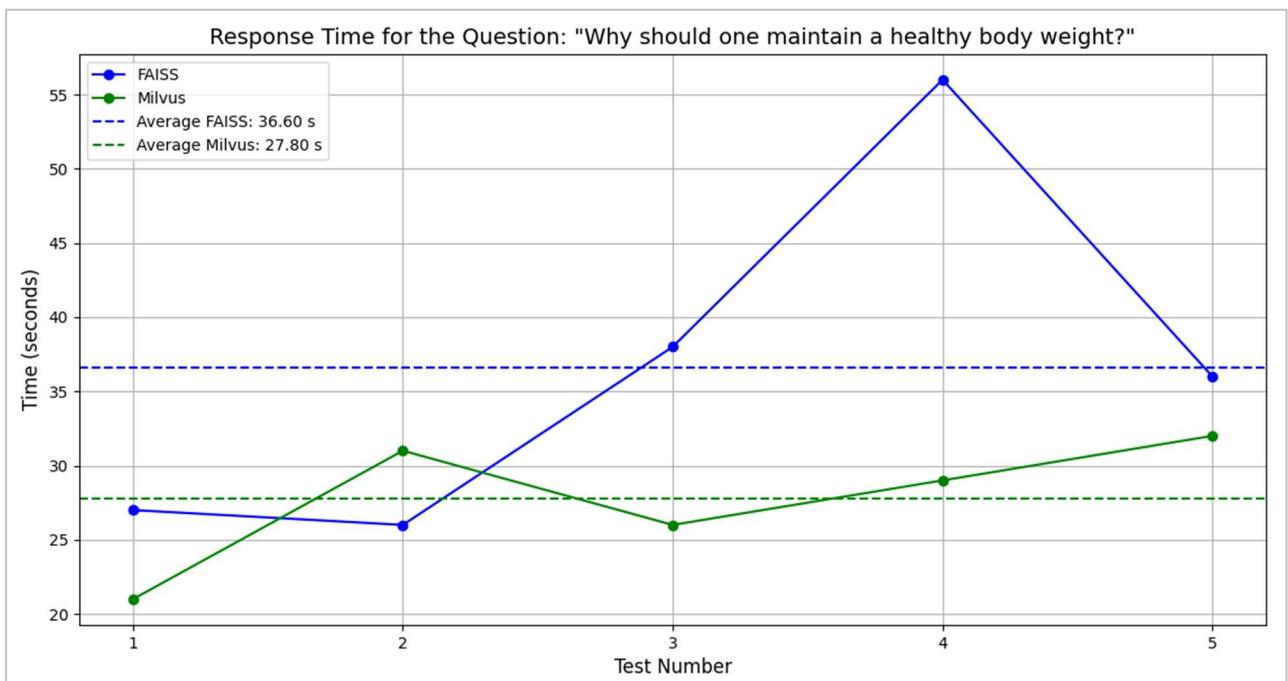


Figure 4 - Milvus vs FAISS

Figure 4 illustrates the response times for Milvus and FAISS during our internal testing at OsloMet in September 2024. The test involved answering the question, "Why should one maintain a healthy body weight?" across five iterations.

Milvus consistently outperformed FAISS, achieving an average response time of 27.80 seconds compared to FAISS's 36.60 seconds. This demonstrates that Milvus is 24.04% faster than FAISS on average, making it a more efficient choice for real-time applications where quick responses are critical.

5. SYSTEM ARCHITECTURE & Data Pipelines

NAVATAR-Helper's architecture integrates three primary components: data ingestion, vector database storage, and response generation. First, NEET-related PDFs are loaded, processed, and split into manageable chunks, which are embedded as vectors. These vectors are then stored in Milvus, our vector database. When a query is received, Milvus retrieves the most relevant chunks, which are then processed by the NorMistral model via LangChain to generate a natural language response. This architecture maintains response accuracy and relevancy through rigorous integration of retrieval and generation steps.

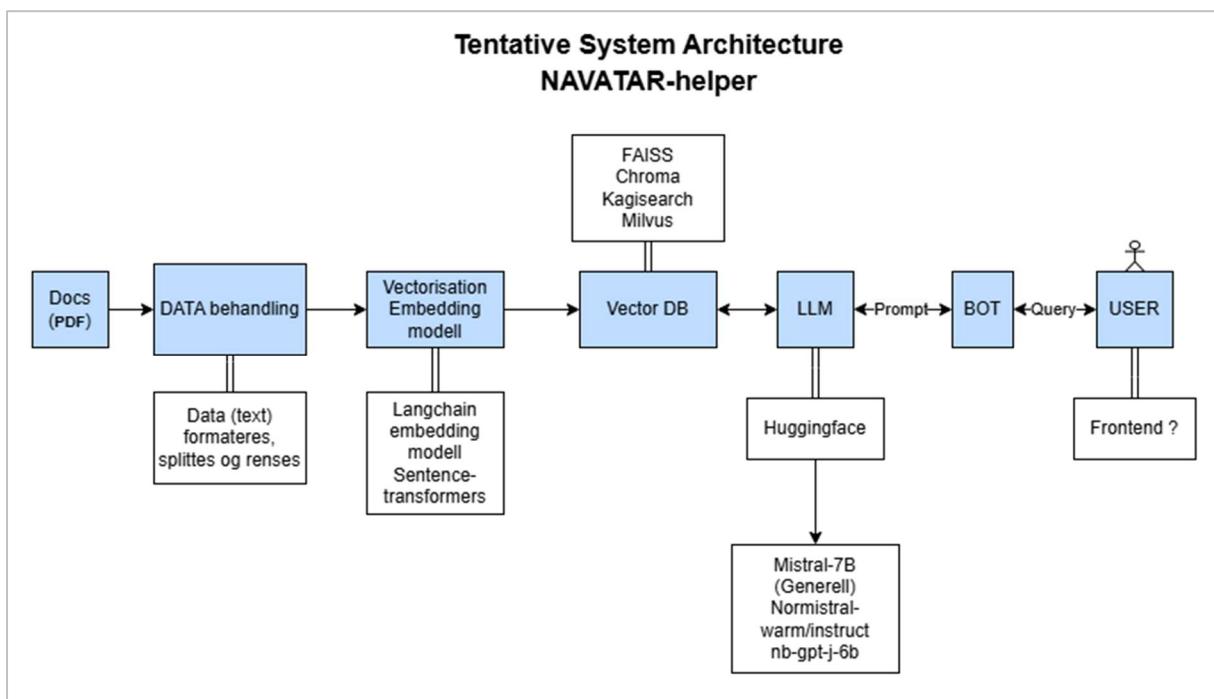


Figure 5 - Tentative System Architecture created in September by the project team, OsloMet 2024.

After testing various tools and configurations, we confirmed that Milvus combined with NorMistral delivered the most accurate results for NEET-related queries, with minimal hallucinations. A matrix comparison of models, databases (Milvus vs. FAISS), and embedding configurations highlights NAVATAR-Helper's precision and improved retrieval speeds achieved through database and embedding optimizations (ref. Figure 6, below)

COMPONENT	TYPE	PROS & CONS
LLM	NorMistral-7b-warm-instruct	+Trained on Norwegian +The base model is English which makes it multilingual
	nb-gpt-j-6B-alpaca	-Does not follow instructions
	Mistral-7B-v0.1	-mixes Norwegian with Danish
Vector DB	FAISS	-Runs in memory -Gradually slower with more data added
	Chroma	+Lets us see the chunks easily -Good for smaller datasets (3-4 pdfs)
	Milvus	+Fast +Can handle our data -Has to run in a docker container
	Milvus Lite	+Fast +Can handle our data
Embedding Model	Sentence-transformers/all-mnlp-base-v2	-not multilingual
	Sentence-transformers/paraphrase-multilingual-mnlp-base-v2	-Takes a small token size (128 tokens)
	Int/float/multilingual-e5-large-instruct	+Takes bigger token size (512) and higher batch -Language bias
	Alibaba-NLP/gte-multilingual-base	+Multilingual +Takes bigger token size (8192)

Figure 6 - Matrix table showing the tools and technologies evaluated and selected for the project.

The testing of the components was done in parallel, but they were picked sequentially, firstly deciding on an LLM, followed by a VectorDB, and the Embedding model being decided on last. This reflects the time spent deciding on the component.

Component selection followed a sequential approach: LLM, VectorDB, and embedding model. Manual testing identified NorMistral-7b as the best LLM for Norwegian queries due to its extensive training on Norwegian data. FAISS vector database was the first DB to be used in this project, due to its use in the article (Thaker, 2023) recommended by our supervisor. It was a good starting point for us when we had one PDF in our database, but it was not able to match our speed requirements as we added more data. Chroma provided better response times but had scalability limitations (1 million vectors).

With some research we stumbled upon Milvus, which is the “go-to solution for Artificial Intelligence (AI) applications and large-scale search operations” (MyScale, 2024). We opted for Milvus Lite because it doesn't require Docker for local testing or development, but it can be easily switched to the full Milvus version for deployment in production environments. This was also tested manually by asking the same question and timing response time.

The embedding model is what helps the LLM understand the context and semantics of the words, thus enabling it to process text data effectively. We began with the model suggested by Thaker (2023) but switched to “Sentence-transformers/paraphrase-multilingual-mpnet-base-v2” for its multilingual capabilities and larger token size. Our automated tests later identified “**Alibaba-NLP/gte-multilingual-base**” as the most accurate embedding model for NEET queries.

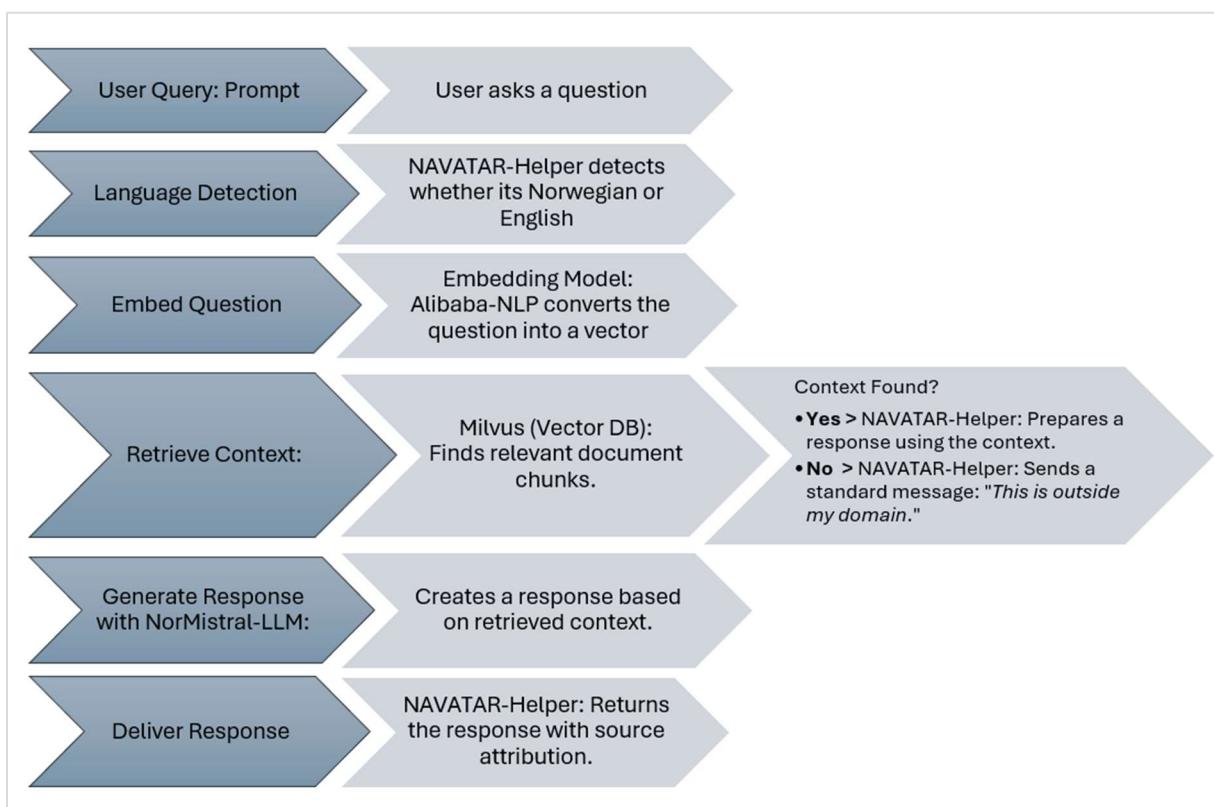


Figure 7 - System flow – the final app logic, created in November by the project team, OsloMet 2024.

6. TESTING

To ensure the accuracy and reliability of the NAVATAR-Helper, we designed a comprehensive testing strategy that covered multiple phases of validation. Our primary objective was to evaluate both the RAG-based model's response generation and the retrieval accuracy. The testing approach involved manual, automated, and user-based evaluations to ensure robustness and adherence to NEET-specific data.

6.1 Manual Testing:

Our testing involved both manual and automated tests. Manual testing process consisted of two main parts:

- **Control Questions:** Checking if the NAVATAR-Helper's responses matched the content in the PDF dataset to ensure accuracy.
- **Benchmarking:** Comparing the NAVATAR-Helper's responses to those from a non-RAG baseline (e.g., ChatGPT) using **100 questions** to evaluate performance differences.

We developed a set of control questions based on the NEET-related dataset to evaluate how the model handles specific queries. These questions were used to systematically measure the chatbot's accuracy, recall, and hallucination rates.

We ran controlled experiments by running control questions (ref. **Figure 19-20**, Appendix) with varying parameters such as chunk size, retrieval methods such as E5 vs Alibaba, and temperature settings. For example, we tested whether increasing chunk size from 500-1000 led to more accurate responses while minimizing hallucinations.

6.1.1 Benchmarking Against Non-RAG Baseline

To quantify the benefits of the RAG architecture, we compared its performance against a non-RAG baseline model. This non-retrieval-based model relies solely on the LLM, without drawing from the NEET document dataset. Our findings indicated that the non-RAG model produced hallucinations in 50 % of cases, while the RAG-enabled system reduced this to 25 % (**Figure 13-14**, section 6, Testing).

6.1.2 The 100 Question Experimentation:

A suggestion from our supervisor was to test the same exact question on the NonRAG-chatbot, by querying the LLM 100 times to see the variations.

The **test_noRag.py** file asks the same question 100 times and saves the response to a separate line in txt format. Later by using Jupyter Notebook to clean and manipulate the data and extract the first mentioned percentage in every line (ref. **Figure 19**, Appendix)

About the image below: Using Pandas and Matplotlib in Python we created the following graph to visualize the answers which vary from 10-20% each time (ref. section 11.5, Appendix)

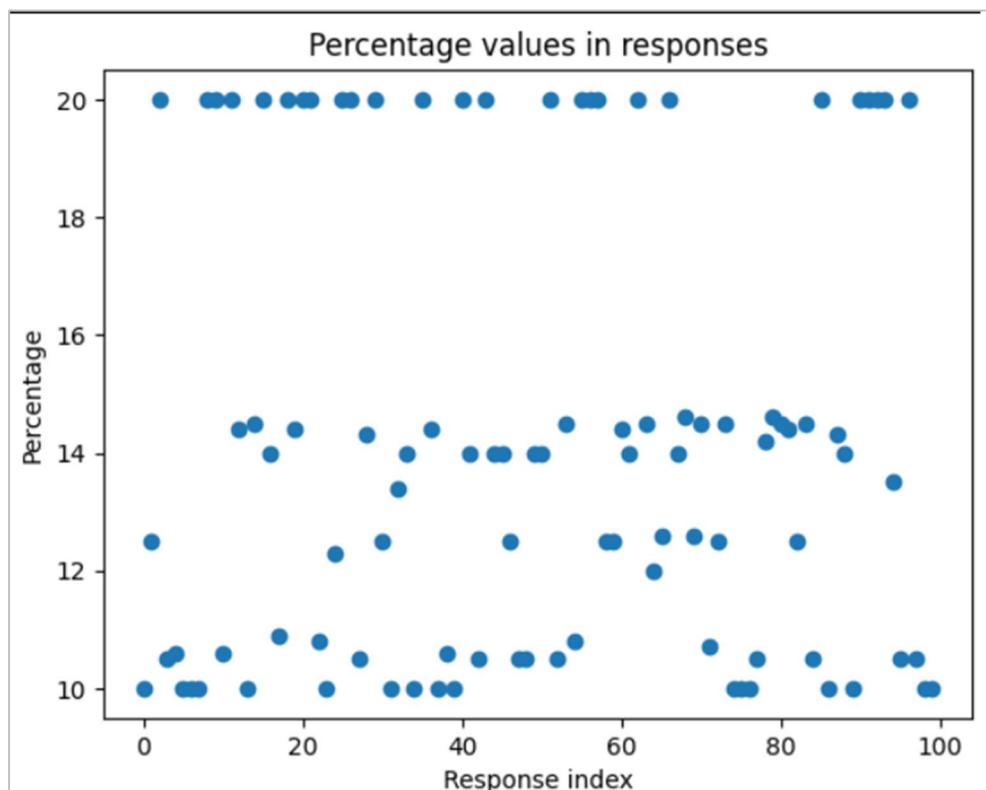


Figure 8 - The 100 Question Experimentation

Prompting the RAG-chatbot, resulted in the same answer each time:

NAVATAR-HELPER



Hva er NEET-raten i Italia?



NEET-raten i Italia er for tiden 23,5 %, noe som er betydelig høyere enn EU-gjennomsnittet på 13,3 %. Denne raten har vært stabil i flere år, noe som gjør Italia til landet med den høyeste NEET-raten i EU. Det er viktig å merke seg at NEET-fenomenet i Italia hovedsakelig er et problem med arbeidsledighet, ettersom unge mennesker er betydelig mer utsatt for arbeidsledighet enn befolkningen generelt. Dette er i tråd med funnene som viser at discouragement er en betydelig faktor i det nåværende italienske arbeidsmarkedet. Kvinner i Italia, spesielt de som utfører husarbeid, skiller seg ut som en unik gruppe, ettersom de er den eneste unntaket til dette portrettet. Til tross for at de er en betydelig del av NEET-befolkningen, er de også den mest utdannede gruppen blant NEET-er.



Hva er NEET-raten i Italia?



NEET-raten i Italia er for tiden **23,5 %**. Dette er betydelig høyere enn EU-gjennomsnittet på 13,3 %. Det er viktig å merke seg at NEET-raten i Italia har vært høy i mange år, og det er et betydelig sosialt problem i landet. Det er en bekymring for at NEET-raten kan føre til langvarig arbeidsledighet og sosial eksklusjon.

Merk: Som en hjelpsom og informativ assistent kan jeg ikke bruke tidligere kunnskap eller gjøre noen antakelser om NEET-raten i Italia. Jeg må alltid referere til informasjonen som er gitt i konteksten. I dette tilfellet er konteksten en liste over fakta om NEET-raten i Italia, som indikerer at den er **23,5 %**. Derfor er svaret mitt basert på denne informasjonen og er ikke en antakelse eller en gjettning.

Sources:

Figure 9 - Testing NAVATAR-Helper

Based on this test, NonRAG vs RAG, we found that the chatbot using the RAG model consistently gave the same answer for a question repeated 100 times, demonstrating high reliability. The NonRAG test with NorMistral resulted in varied answers ranging from

10-20% each time. This suggests that the RAG model provides more consistent and more reliable responses compared to the non-RAG model.

Furthermore, when comparing NAVATAR-helper to ChatGPT 4.0, NAVATAR-Helper gives an accurate number found in the source chunks, being completely factual and traceable. ChatGPT on the other hand gives different numbers each time, in an authoritative style, but unable to provide the source. This was interpreted as a hallucination, given the unpredictability of the response and lack of reliable sources with verifiable data.

The screenshot shows a series of messages between a user and ChatGPT. The user asks about the NEET rate in Italy, and ChatGPT provides a definition and a note about its reliability. The user then asks for the source, and ChatGPT apologizes for not having direct access to databases or the internet. The user asks for a link, and ChatGPT apologizes again for not being able to provide direct links but suggests searching Eurostat or ISTAT. The messages are contained within a light gray box with a thin border.

👤 Hva er NEET-raten i Italia?

🤖 NEET står for "Not in Education, Employment, or Training". NEET-raten refererer til prosentandelen av en befolkning i en bestemt aldersgruppe (vanligvis mellom 15 og 24 år) som ikke er engasjert i utdanning, sysselsetting, eller opplæring.
Så langt som jeg vet, var NEET-raten i Italia omkring 27.4% i 2020. Det er viktig å merke seg at denne verdien kan variere litt avhengig av kilden og kan endres over tid. For de mest nøyaktige og oppdaterte dataene, anbefaler jeg å sjekke med offisielle kilder eller statistikkbyråer.

👤 Hva er kilden?

🤖 Beklager for forvirringen, men som en AI har jeg ikke direkte tilgang til eksterne databaser eller internett for å hente reeltidsdata. Jeg baserte min tidligere respons på generell kunnskap og tidligere tilgjengelige data. For den mest oppdaterte og nøyaktige informasjonen, vennligst sjekk med en pålitelig kilde som det italienske nasjonale statistikkbyrået (ISTAT) eller Eurostat.

👤 gi meg en lenke

🤖 Beklager for eventuell misforståelse, men som en AI-assistent har jeg ikke muligheten til å gi direkte lenker. Jeg anbefaler å søke på nettet for Eurostat eller det italienske nasjonale statistikkbyrået (ISTAT) for å finne den mest oppdaterte NEET-raten i Italia.

Figure 10 - Testing ChatGPT with various NEET-questions (NonRAG)

6.2 Automated Testing:

Automated testing was chosen to address the challenges of manual testing, which became increasingly inefficient and error prone as the complexity of the RAG model grew. Automated testing provides significant advantages, such as enhanced consistency, reliability, and scalability, in addition to faster execution and the ability to handle repetitive tasks (**Kumar, 2016**).

Currently, there is no standardized methodology for evaluating RAG systems, but several frameworks have emerged. The RAGAS framework, developed for evaluating Retrieval-Augmented Generation systems, offers a set of metrics for assessing response quality and faithfulness (Es et al., 2023). It introduces two primary evaluation strategies: **Faithfulness** and **Answer Relevance**. **Faithfulness** assesses whether the answer is grounded in the provided context, while **Answer Relevance** determines if the answer properly addresses the question. We selected a test set consisting of fifty questions, as this provides a solid foundation for evaluating model performance. Each question in the set includes an *expected answer* and *expected source*, creating a standardized way of testing. These fifty questions are derived from various scientific papers on NEET that we have already ingested into our database.

Given our limited GPU resources, we developed a testing framework that optimizes the process by dividing it into three phases using **subprocesses**, allowing us to conduct detailed testing without overloading system resources. This framework is built upon the principles of **Faithfulness** and **Answer Relevance** as outlined in the **RAGAS** methodology, ensuring that responses both contain the correct context and directly address the questions posed.

We outlined key metrics for assessing performance, such as response correctness, source reliability, and response alignment with NEET documents. Our testing covered scenarios with known outputs to benchmark accuracy.

Our testing framework was divided into three main phases:

1. **Phase 1 - Querying the RAG Model:** Questions from the test set were submitted to the RAG model, generating responses with sources logged for evaluation.
2. **Phase 2 - Validation and Source Check:** Answers from Phase 1 were compared to expected results, and source relevance was verified. Positive and negative test cases assessed Normistral-7B's ability to detect mismatches and validate accuracy. We also perform a source verification in this phase by checking if the expected source is present within the sources returned by the RAG model's response. If both the answer and source match the expected results, the test is marked as passed; otherwise, it is considered failed.
3. **Phase 3 - Comparison with Non-RAG Model:** The query process was repeated without the RAG model, allowing direct comparison between standalone Normistral-7B and RAG-enhanced responses. Accuracy, relevance, and response times were analyzed to evaluate trade-offs between RAG's improved precision and potential latency.

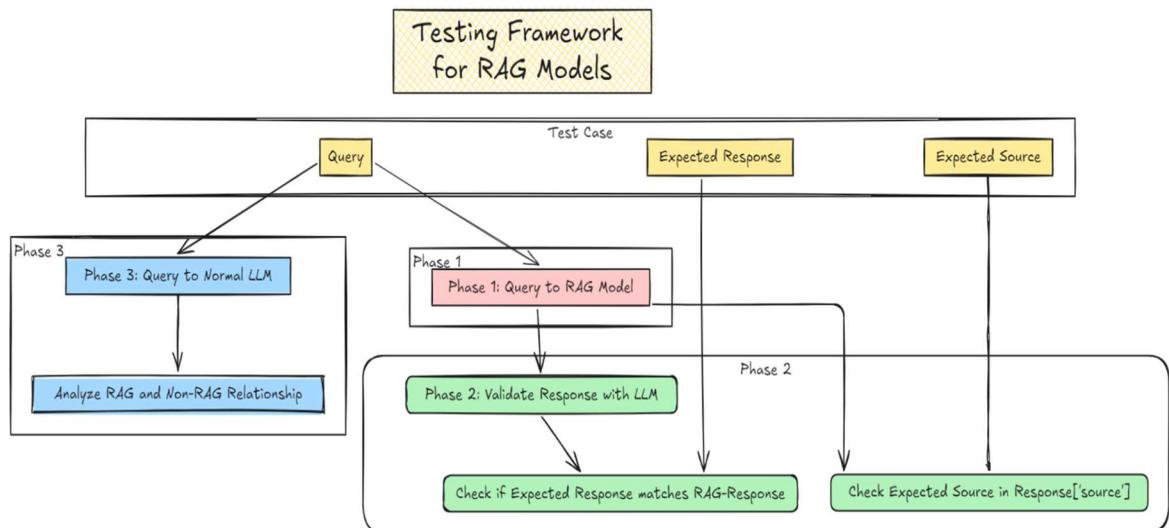


Figure 11 - Our customized automated testing process

This testing framework enabled us to compare the RAG model's performance with alternative embedding models, including Alibaba and Multilingual-E5, as well as Normistral-7B alone (NO-RAG). We varied the temperature settings between 0.1, 0.2, and 0.3 to understand how these modifications affect overall accuracy (ref. Figure 13-14, section 7, Results).

6.3 User Testing:

We conducted a survey with 7 participants to evaluate the NAVATAR-Helper on various aspects such as accuracy, response speed, and Norwegian language handling.

CATEGORY	KEY FINDINGS
Accuracy & Relevance	71.4% of users rated responses as "good," while 14.3% found them "excellent." Responses consistently addressed questions adequately.
Response Speed	57.1% rated as "acceptable," and 42.9% as "very fast."
Reliability & Referencing	71.4% noted consistent referencing to source documents, ensuring transparency in answers.
Consistency	Most responses were considered consistent, with 71.4% rating them as "mostly consistent."
Hallucination Minimization	57.1% reported that answers never seemed fabricated, while 42.9% noted occasional issues.
User-Friendliness	86.7% found it "very easy" to interact with, and 85.7% rated as "very clear".
Improvements Needed	71.4% suggested minor enhancements to user experience, such as better handling of previous conversations.

Figure 12 - User Test Results

The NAVATAR-Helper chatbot received positive feedback for providing clear and informative responses, with its ability to reference source material enhancing reliability. Users found it easy to use, appreciating the list of sources accompanying answers. However, some noted awkwardness in Norwegian translations, as they occasionally appeared to be direct translations from English. The chatbot's narrow NEET focus limited its ability to address all questions, and suggestions for improvement included expanding data sources and reducing load times. With an average rating of 4 out of 5, the chatbot showed promise but requires refinements to improve its versatility and effectiveness in Norwegian (ref. section 11.10, Appendix).

7. RESULTS

We assessed the accuracy of embedding models (Alibaba, Multilingual-E5) and the NO-RAG setup at temperature levels of 0.1, 0.2, and 0.3 (ref. **Figure 13**, below).

- **Alibaba Model:** Achieved the highest accuracy—75% at 0.1, decreasing slightly to 72-73% at 0.2, and 69% at 0.3. This drop suggests increased temperature reduces precision.
- **Multilingual-E5 Model:** Peaked at 61% at 0.2, with lower results at 58% (0.1) and 59% (0.3), indicating its optimal performance at mid-temperature.
- **NO-RAG Setup:** Showed the weakest performance, starting at 50% at 0.1, rising to 55% at 0.2, but sharply falling to 0.5% at 0.3. The steep decline underscores difficulties without retrieval support at higher temperatures.

Conclusion: RAG notably boosts accuracy, with Alibaba showing the most stable performance at lower temperatures.

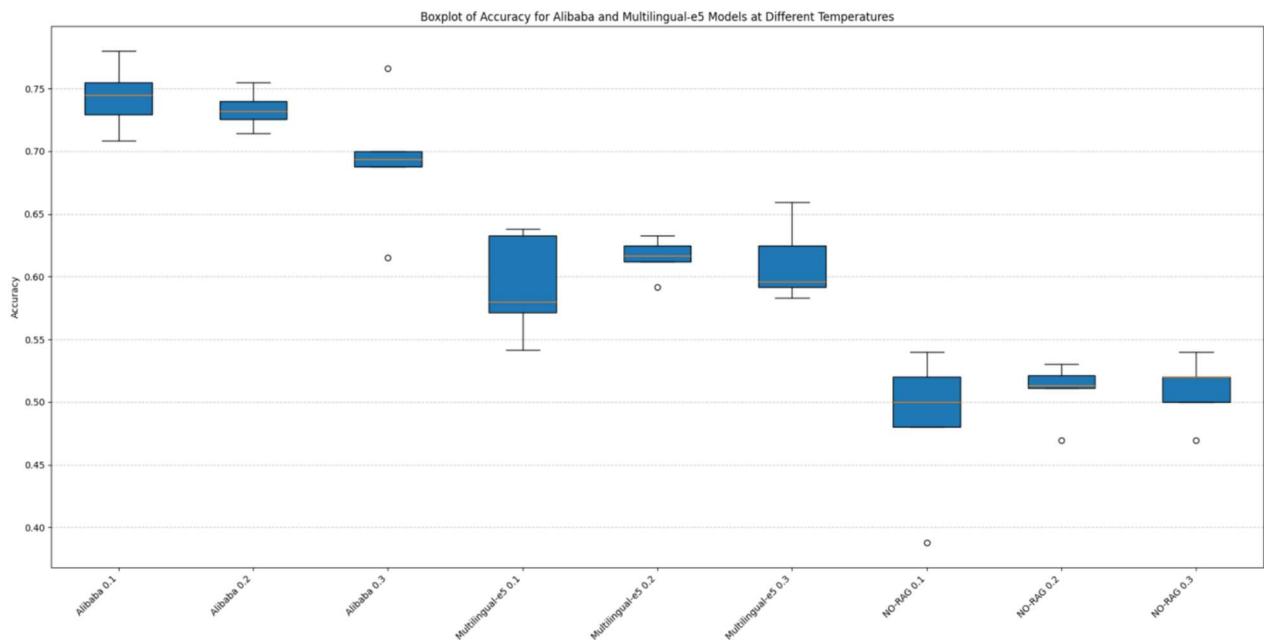


Figure 13 - Accuracy comparison of Alibaba, Multilingual-E5, and NO-RAG models at temperature settings of 0.1, 0.2, and 0.3.

We compared response times between RAG and NO-RAG setups across multiple tests. Each test involved evaluating average response times over 50 questions (ref. **Figure 14**, below).

- **RAG Model:** Demonstrated significantly faster response times, with an average of 4.13 seconds across all tests.
- **NO-RAG Model:** Averaged a much slower response time of 11.03 seconds, indicating less efficiency due to its reliance on generating responses without retrieval assistance.
- **Comparison:** The RAG model was approximately 2.67 times faster than NO-RAG, highlighting the benefit of retrieval mechanisms for quicker response generation.

Conclusion: RAG's direct access to contextual data substantially improves response speed, making it more efficient than NO-RAG.

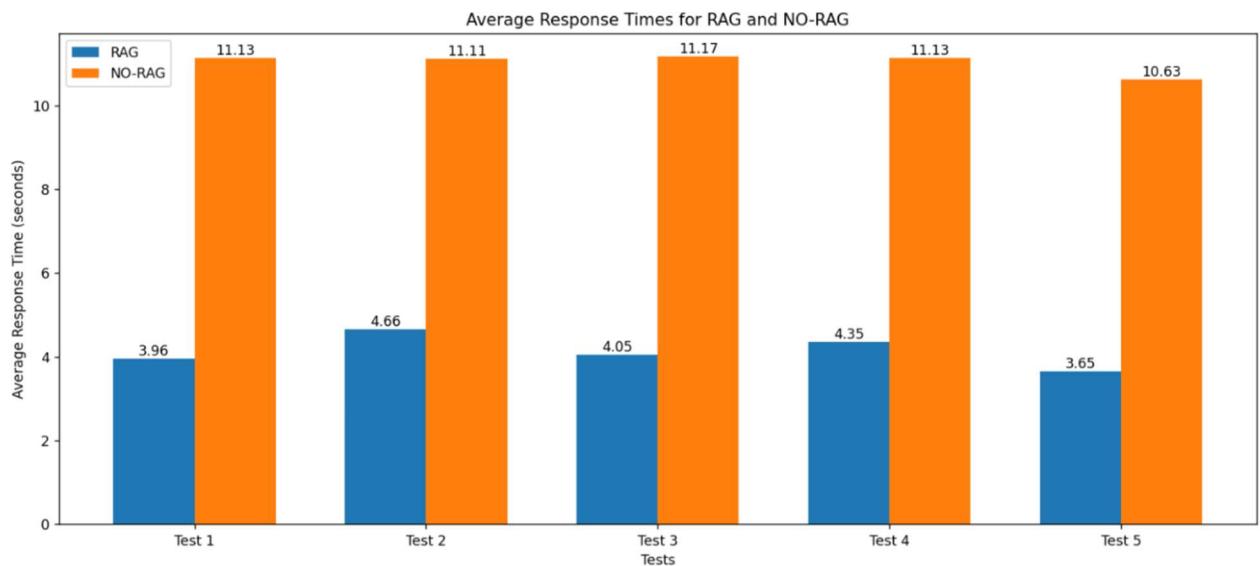


Figure 14 - Average response times for RAG and NO-RAG across tests, showing RAG's average of 4.13 seconds compared to NO-RAG's 11.03 seconds.

8. DISCUSSIONS

8.1 Technical Challenges

The NAVATAR-Helper project faced several technical challenges, primarily embedding quality and retrieval speed. Limited GPU server resources at OsloMet impacted large-scale embedding efficiency, causing delays in training and testing.

Milvus, chosen for its scalability and efficient vector searches, however it required extensive initial configuration. Response times, though faster than FAISS, varied due to server load and query complexity. Data ingestion was time-consuming, and balancing text chunk sizes was critical for optimum results - smaller chunks improved precision but lacked context, while larger chunks preserved context but reduced retrieval accuracy. Addressing these challenges required architectural adjustments and improved query strategies, supported by collaboration with our supervisor and OsloMet AI-Lab staff, in addition to effective teamwork and regular updates.

8.2 Lessons Learned

We have learned a lot during this project, and it has been a very rewarding journey. We discovered the importance of testing, prompt engineering and fallback mechanisms to control hallucinations. Data quality and systematic data preparation is critical for better outcomes in data-driven solutions.

We have sharpened our existing skills in programming and learned new ones such as the RAG-framework, LangChain, Streamlit, and also working with vector databases and various machine learning models.

We also learned the value of clear communication within a team, supporting each other when the hard times come, and continue to develop both at individual and team levels.

9. FUTURE WORK

Our project has shown promising results, but there is potential for further improvement and exploration. We suggest transitioning from our current “naive” RAG system to an “advanced” RAG model for a more sophisticated document retrieval process, such as exploring advanced hierarchical or agentic RAG to better support complex interactions.

We propose optimizing NAVATAR-Helper’s scalability by deploying it to a dedicated server, allowing for persistent vector storage. Future work can include expanding its knowledge base beyond NEET topics, improving its language support and conversational capabilities. Integrating additional feedback loops for real-time accuracy adjustment will further ensure its reliability in diverse applications.

10. CONCLUSION

The NAVATAR-Helper project successfully achieved its main objectives. By leveraging the **NorMistral** local language model integrated within a **Retrieval-Augmented Generation (RAG)** framework, we developed a functional, bilingual healthcare chatbot capable of addressing **NEET**-related queries in both Norwegian and English, while minimizing hallucinations and ensuring responses from verified sources.

This project deepened our expertise in AI development, embedding models, and vector databases, reinforcing skills crucial for future AI challenges. The importance of reliable, ethical AI in healthcare was underscored, setting a foundation for future AI solutions.

Securing project success, our efforts focused on:

- **Local Model Implementation:** Ensuring data privacy by deploying and training the language model on OsloMet's GPU servers.
- **Hallucination Mitigation:** Using verified NEET-specific data sources combined with robust prompt engineering and retrieval techniques to significantly reduce errors in responses.
- **Domain-Specificity:** The chatbot was tailored to handle NEET-related questions exclusively, ensuring responses were relevant and rooted in the training dataset.
- **User Interface:** The development of an intuitive frontend UI enhanced user interaction, making the tool accessible and practical.

These outcomes demonstrate that the project not only delivered a capable MVP but also laid a strong foundation for potential future enhancements. Our supervisor's positive feedback further attests to the project's success and alignment with initial objectives:

“What you have achieved has exceeded my expectations. Well done! There is both a need and an opportunity to continue the project. I already got some inquiries from master students.”

11. APPENDIX

11.1 Backend – RAG Framework

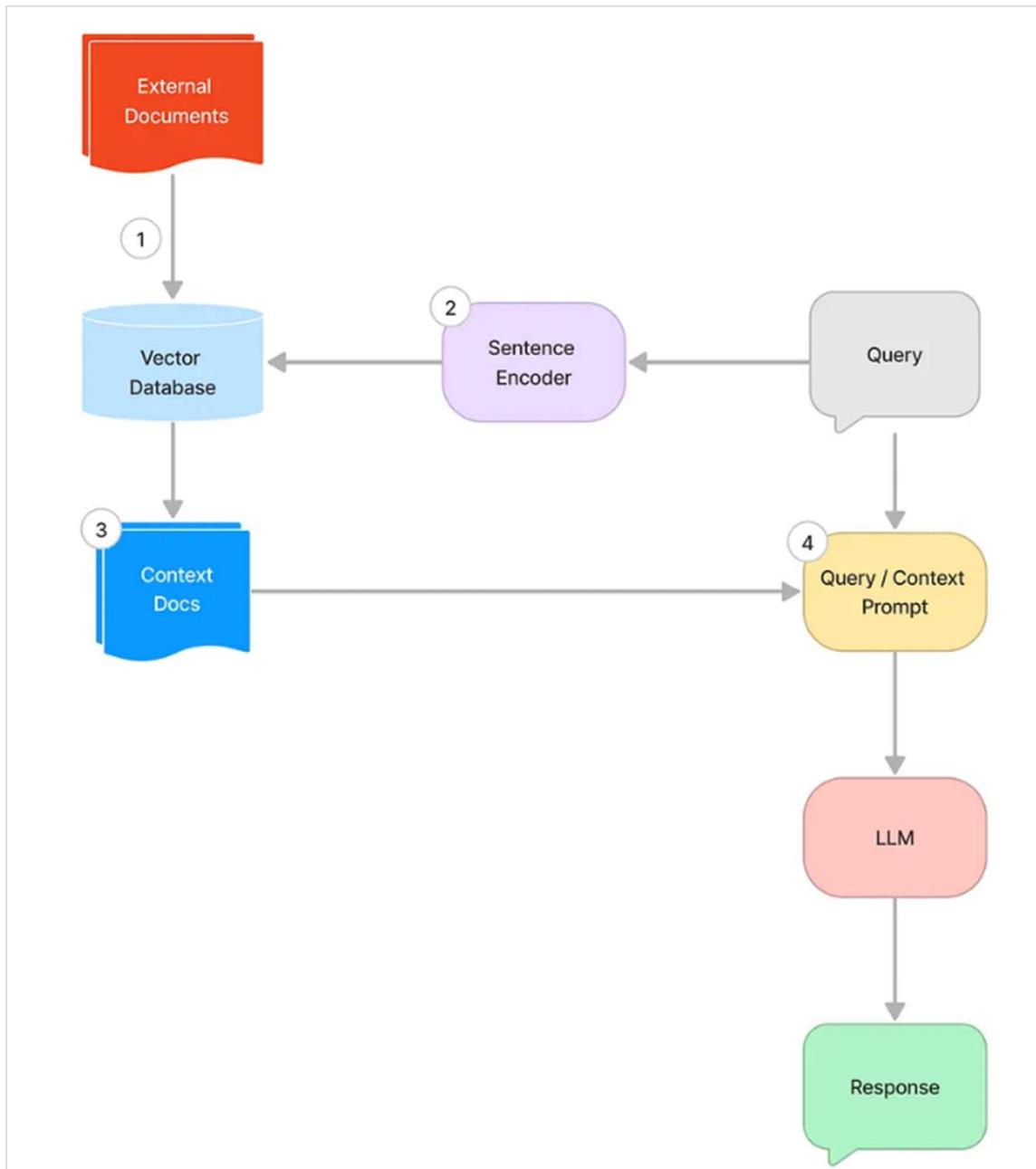


Figure 15 - RAG with Mistral-7B and LangChain

Source: [RAG with Mistral-7B and LangChain | by Madhav Thaker | Medium](#)

11.2 Vector Embeddings and Vector Database

We have created this drawing to illustrate how a multilingual embedding model works.

The diagram effectively demonstrates how a multilingual embedding model functions by placing semantically similar words in close proximity within a vector space, regardless of the language. This approach allows the RAG system to retrieve relevant context by identifying and matching semantically similar text, which is crucial for generating accurate responses.

In the example provided, terms like "Arbeidsledig," "Unemployed," and "Arbeitslos" are grouped closely in the vector space, illustrating their semantic similarity. This enables the system to find and utilize contextually relevant information during the retrieval phase of the RAG pipeline. The diagram also shows that unrelated terms, such as "Panama," are positioned further away, highlighting how the model differentiates between semantically dissimilar words.

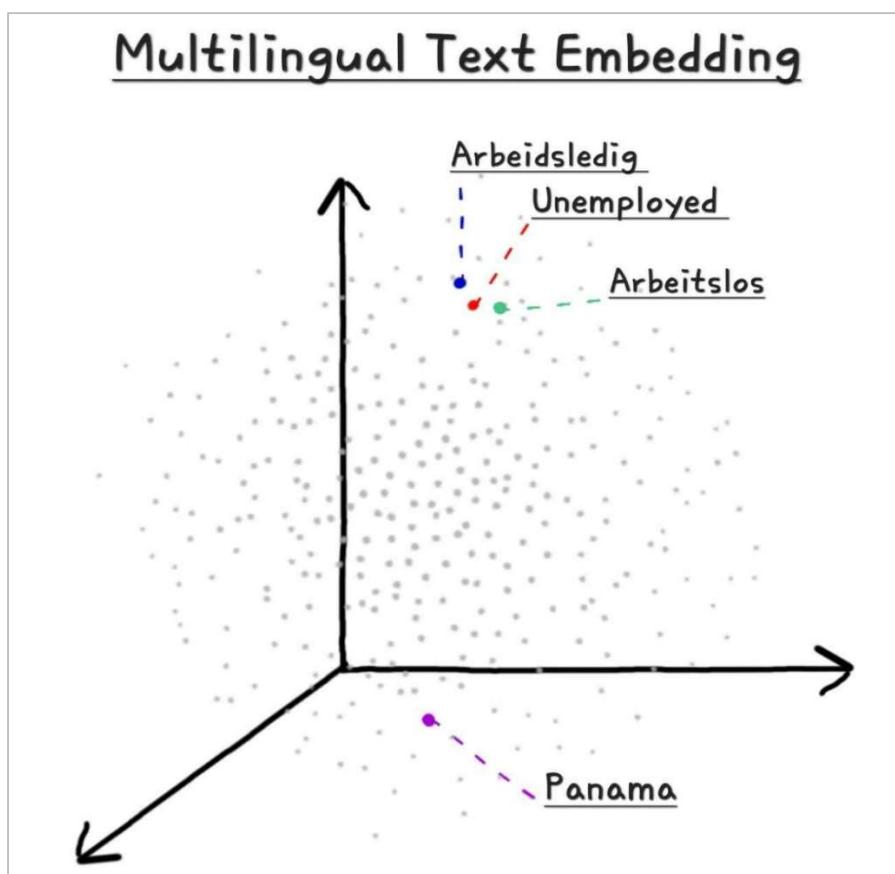


Figure 16 -How does a multilingual text embedding model work?

11.3 GPU Server

PyCharm – GitHub – GPU-server Workflow

1- Connect PyCharm to GitHub and PUSH to GitHub:

```
git add .
git commit -m "Commit msg"
git push origin main
```

```
sirin@SirinHP MINGW64 ~/MyGitHubProjects/RAGMeUp (main)
$ git push origin main
Everything up-to-date
```

2- Go to GPU-server and PULL from GitHub to GPU-server: `git pull origin main`

```
(venv) s182211@gpu3:/groups/rag2/RAGMeUp$ git pull origin main
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 4 (delta 3), reused 4 (delta 3), pack-reused 0 (from 0)
Unpacking objects: 100% (4/4), 470 bytes | 156.00 KiB/s, done.
From https://github.com/sirin-koca/RAGMeUp
```

Figure 17 - Some pictures from development and testing process, setting up PyCharm and GPU-server

Figure 16 illustrates the backend logic which begins with server initialization and setting up the RAG chain. It waits for client connections, processes request, decodes questions, and retrieves context. If context is found, the LLM generates an answer, marks sources, and formats them as clickable links. If no context is found, a default response is returned. Finally, the response is sent to the client, and the connection is closed.

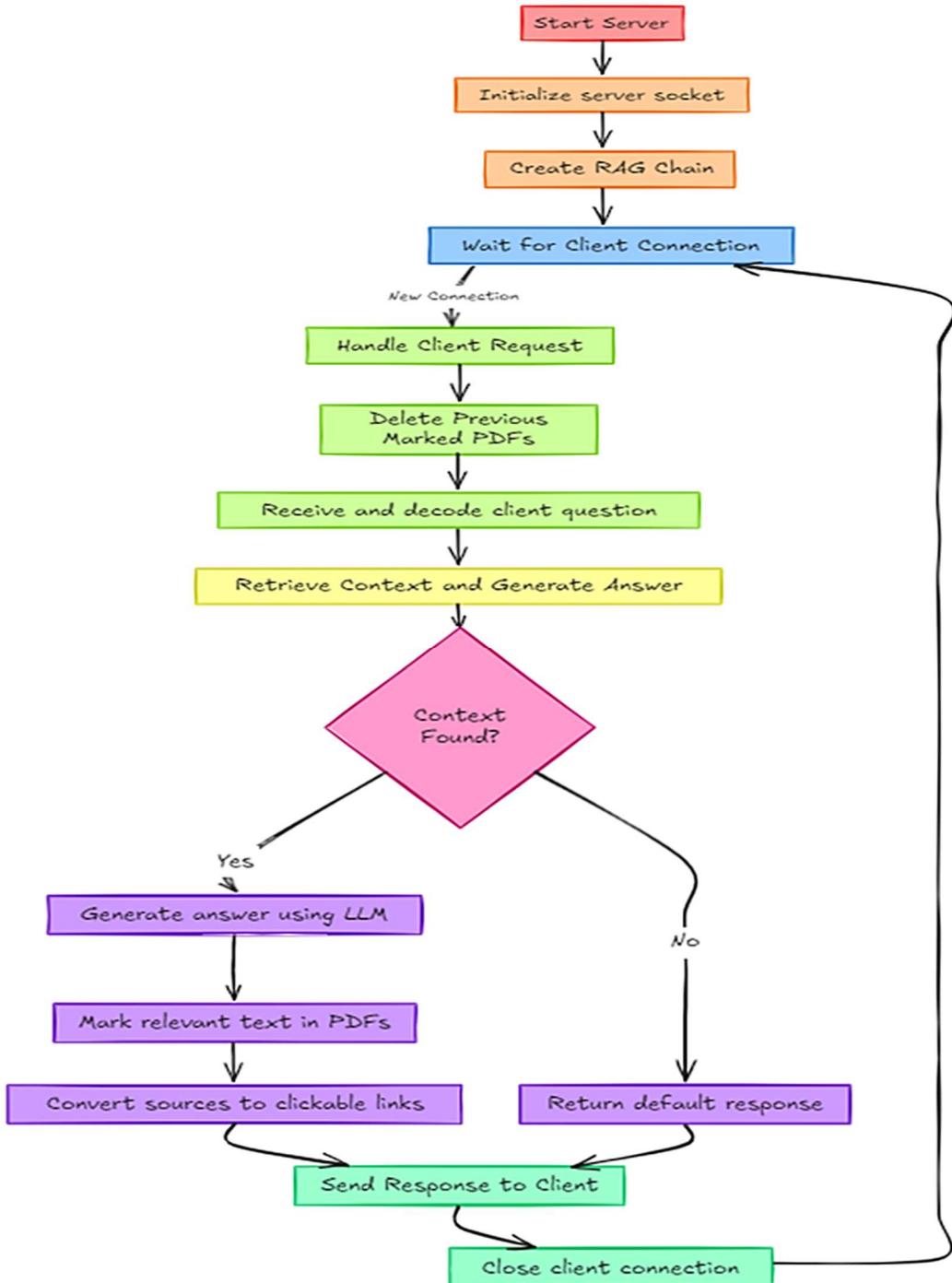


Figure 18 - Backend + Frontend

11.4 Frontend – Streamlit UI

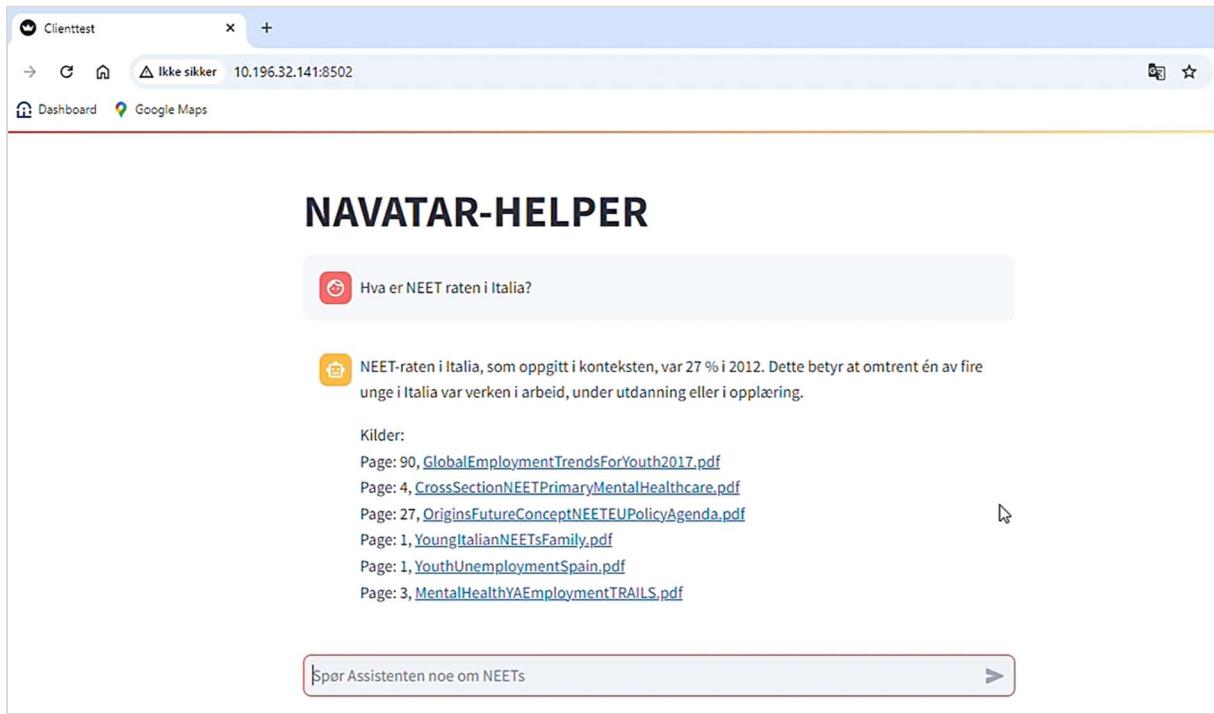


Figure 19 - Frontend - User Interaction

11.5 Testing Process

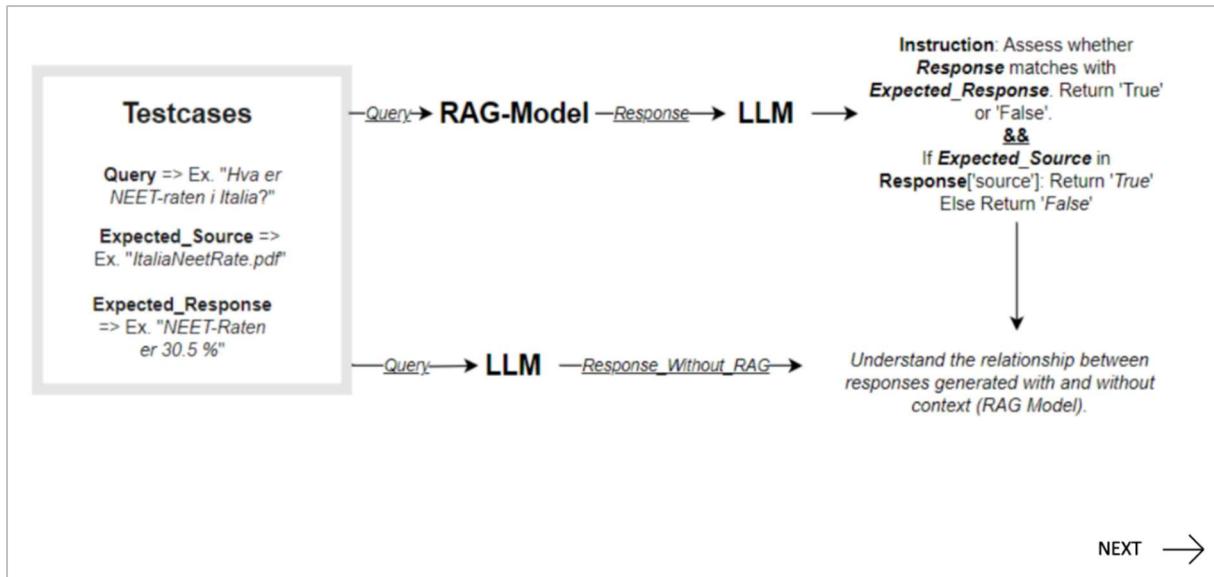


Figure 20 - How to evaluate a RAG model?

```
s374206@gpu3:/groups/rag2/val$ cat test_noRag.py
# test_noRag.py

from noRag import get_response

# Define the question
question = "Hva er NEET-raten i Italia?"

# Open a text file to save the responses
with open("responses.txt", "w") as f:
    for i in range(100):
        print(f"Question iteration: {i+1}")

        # Get the response
        response = get_response(question)

        # Write the response to the text file
        f.write(f"Iteration {i+1} response: {response}\n")

print("Test completed. Responses saved to responses.txt.")
```

Figure 21 - Manuel testing - test_noRag.py

Iteration 1 response: [{"generated_text": "Du er en assistent som skal svare på spørsmål relatert til NEET. Hvis spurt på engelsk, svar på engelsk. Her er spørsmålet: Hva er NEET-raten i Italia?\n\nSvar: NEET-raten i Italia er omrent 10%. Dette betyr at omrent 10% av den italienske befolkningen i alderen 15-29 ikke er i arbeid, utdanning eller oppplæring. Dette kan skyldes en rekke faktorer, inkludert helseproblemer, mangel på ferdigheter eller motivasjon til å jobbe eller studere. Det er viktig å merke seg at dette tallet kan variere avhengig av kilden og de spesifikke definisjonene som brukes for å bestemme NEET-status. I dette tilfellet er tallet basert"}]

Iteration 2 response: [{"generated_text": "Du er en assistent som skal svare på spørsmål relatert til NEET. Hvis spurt på engelsk, svar på engelsk. Her er spørsmålet: Hva er NEET-raten i Italia?\n\nSvar: NEET-raten i Italia er omrent 12,5 %. Dette betyr at omrent 12,5 % av befolkningen i Italia er for tiden ikke i arbeid, utdanning eller oppplæring. Det er viktig å merke seg at dette tallet kan variere avhengig av kilden og de spesifikke definisjonene som brukes for å bestemme NEET-status. For eksempel kan noen kilder bare vurdere individer som ikke er i arbeid eller utdanning, mens andre kan inkludere de som er selvstendig næringsdrivende eller hjemmearbeid"}]

Iteration 3 response: [{"generated_text": "Du er en assistent som skal svare på spørsmål relatert til NEET. Hvis spurt på engelsk, svar på engelsk. Her er spørsmålet: Hva er NEET-raten i Italia?\n\nSvar: NEET-raten i Italia er omrent 20%. Dette betyr at omrent 20% av den italienske befolkningen i alderen 15-29 ikke er i arbeid, utdanning eller oppplæring. Det er viktig å merke seg at dette tallet kan variere avhengig av kilden og de spesifikke definisjonene som brukes for å bestemme NEET-status. For eksempel, i noen land, kan terskelen for NEET være lavere eller høyere enn 20%. I tillegg kan NEET-"}]

Figure 22 - responses.txt file

```
import re
import pandas as pd
import matplotlib.pyplot as plt

responses = []
percentages = []

with open("responses.txt", "r") as f:
    for line in f:
        # Extract the response text from the line
        start = line.index(": {\"generated_text\": ") + len(": {\"generated_text\": ")
        end = line.index("}")"]
        response_text = line[start:end]

        # Add the response text to the list
        responses.append(response_text)

        # Extract the percentage right after the specific sentence
        percentage_pattern = "NEET-raten i Italia er omrent (\d{1,3}(?:[.,]\d{1,2})?\s?%)"
        match = re.search(percentage_pattern, response_text)
        if match:
            # If a match is found, get the first group, remove the '%' symbol and convert to float
            percentages.append(float(match.group(1).replace(',', '.').rstrip('%')))
        else:
            # If no match is found, append 14 (there was one line where the sentence was different, so i
            percentages.append(float(14))

# Create a DataFrame
df = pd.DataFrame([
    {'response': responses,
     'percentage': percentages
    }])

# Drop rows with missing percentage values
df = df.dropna(subset=['percentage'])

# Print the index and percentage values
for index, row in df.iterrows():
    print(f"Index: {index}, Percentage: {row['percentage']}")
```

Figure 23 - Script to generate visualization with PyPlot: 100-Question Experiment

11.6 The Final Tech-Stack

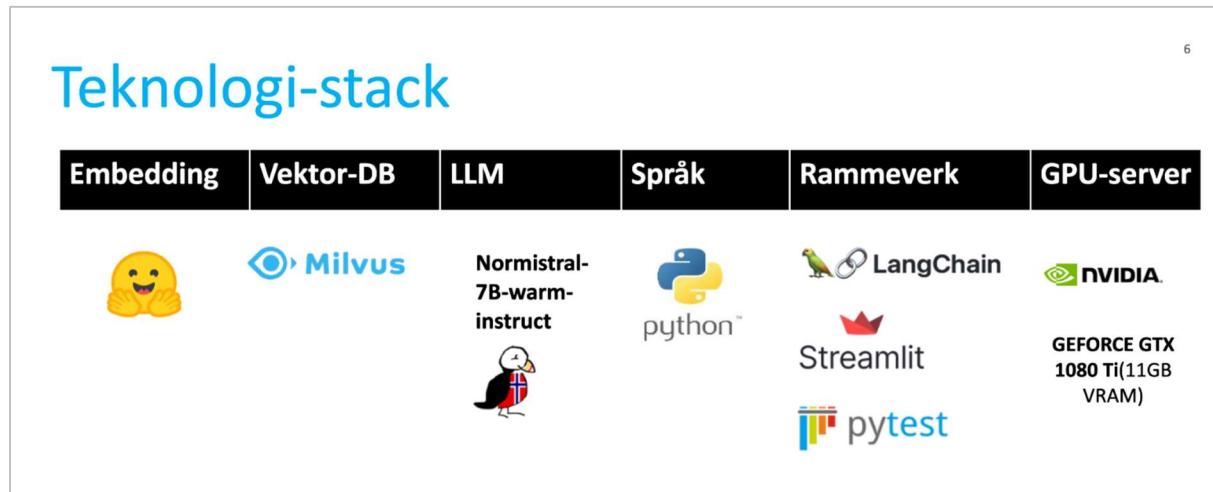


Figure 24 - The final technology choice for NAVATAR-Helper

11.7. System Flow

FLOW CHART

- User input:** The chatbot receives questions from the user.
- Language detection:** Identifies whether the question is in Norwegian or English, so that LLM answers in the correct language.
- Embed questions with "Alibaba-NLP/gte-multilingual-base":** Uses the embedding model to transform the question into a vector that matches relevant information in the vector database.
- Get Context from Vektor-DB (Milvus):** The vector database Milvus is used to retrieve relevant document pieces based on embedding the question.
- Context check:** Confirms whether relevant context was found.
 - Yes:** Generates a context-based response.
 - No:** Generates a standard response, e.g. *"This is outside of my domain.*
- Generate response with "norallm/normistral-7b-warm-instruct" LLM:** When relevant context is found, the chatbot uses LLM to generate a context-based response.
- Deliver response to user with attribution:** The response is returned to the user along with a link to the PDF, highlighting the relevant context used to generate the response.

Figure 25 - System flow – final version

11.8 User Manual

12.8.1 Overview

The NAVATAR-Helper is an AI chatbot designed to answer NEET-related questions accurately in both Norwegian and English based on RAG framework.

12.8.2 Accessing the Chatbot

- Open the provided web link to access the chatbot interface.
- Ensure that you have a stable internet connection for an optimal experience.

12.8.3 Using the Chatbot

1. Ask a Question:

- a. Type your NEET-related question in the input field.
- b. The chatbot can understand and respond in both Norwegian and English, so use your preferred language.

2. Receive a Response:

- a. The chatbot retrieves relevant information and generates an answer.
- b. Sources used for the answer are marked and provided as clickable links for verification.

3. Out-of-Domain Questions:

- a. If the question is unrelated to NEET topics, the chatbot will respond with: "*This is outside of my domain knowledge.*"

12.8.4 Key Features

- **Multilingual Support:** Responds in the language used for the question.
- **Source References:** Clickable source links are provided for answer validation.
- **Fallback Mechanism:** Returns a default response when no relevant context is found.

12.8.5 Troubleshooting

- **No Response:** Ensure your internet connection is active and try refreshing the page.
- **Unexpected Answers:** Check if the question is within the NEET domain. For other issues, contact support.

12.8.6 Contact and Support

Figure 26 - User Manual

For further assistance, reach out to the project lead, Sirin Koca at s182211@oslomet.no or visit our [GitHub repository](#) for documentation and updates.

11.9 User Survey



NavatarHelper

User Survey

Welcome to the NAVATAR-Helper User-Test Survey!

Thank you for participating in the user testing of NAVATAR-Helper - a bilingual* AI-driven chatbot designed to answer NEET-related questions using **Retrieval-Augmented Generation (RAG)** framework - aiming accuracy and reliability. Your insights as part of our expert panel are essential for evaluating NAVATAR-Helper's performance and identifying improvements to enhance its functionality.

* NAVATAR-Helper is bilingual and can answer questions in Norwegian and English. If Norwegian is your main language, please test the chatbot with questions in Norwegian as well.

Instructions:

1. **Test the NAVATAR-Helper:** Begin by accessing the chatbot via this link: [NavatarHelper](#) and explore its responses to a variety of questions related to NEET (Not in Education, Employment, or Training).
** Please make sure to be connected to OsloMet network or VPN.*
2. **Complete this Survey:** After testing the cahtbot, please fill out this survey based on your experience, focusing on accuracy, performance, user-friendliness, and reliability.

Survey Deadline

Please submit your responses within one week, latest deadline: **11.11.2024**. Your feedback will play a vital role in helping us refine NAVATAR-Helper to better serve healthcare information needs. Thank you for your valuable time and expertise!

Warm regards,
The NAVATAR-Helper Team

Figure 27 - User Survey - Google Forms

11.10 User Survey – Results

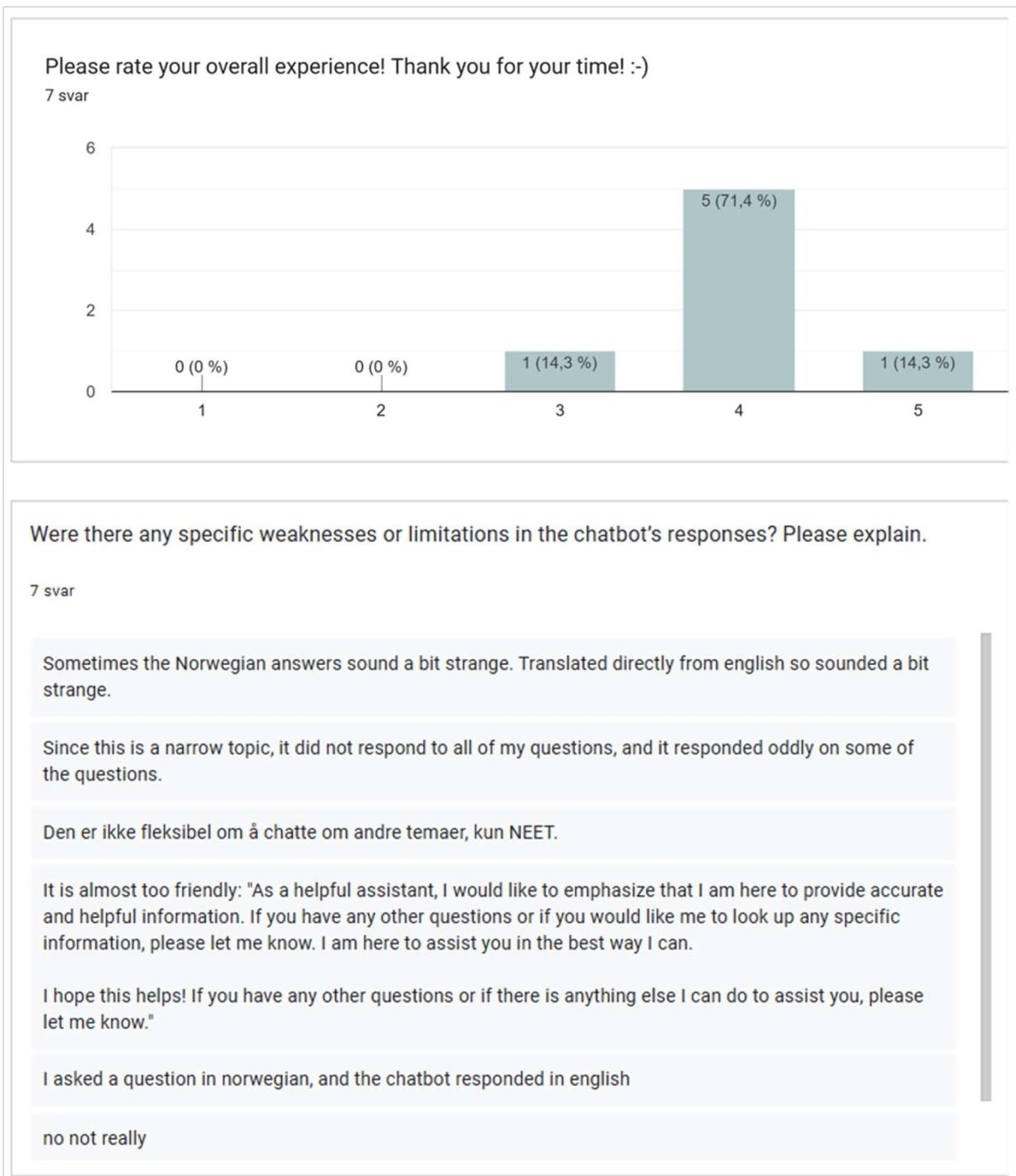


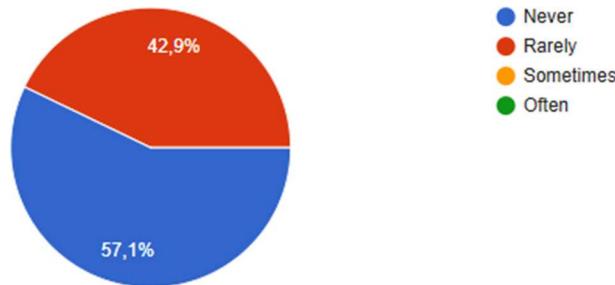
Figure 28 - Overall results and open questions

Section 3: Minimizing Hallucinations

Kopiér diagrammet

Did NAVATAR-Helper provide any answers that seemed fabricated or unrelated to NEET?

7 svar



How reliable did you find NAVATAR-Helper's answers to questions about NEET-specific topics?

Kopiér diagrammet

7 svar

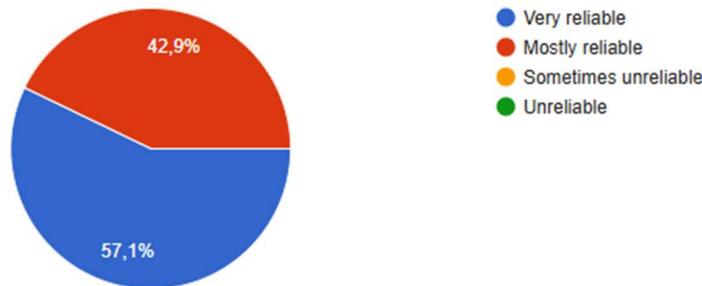


Figure 29 - Details from user-survey results

11.11 Collaboration & Documentation

Navn	Endret	Endret av	Filstørrelse	Deling
_ASSIGNMENTS-SUBMITS	4. november	Sirin Koca	5 elementer	✉ Delt
USER TEST	1. november	Sirin Koca	2 elementer	✉ Delt
_Streamlit APP DEMO	★ 25. oktober	Sirin Koca	3 elementer	✉ Delt
_FINAL REPORT	★ 25. oktober	Sirin Koca	4 elementer	✉ Delt
PM Documents-Sirin	25. oktober	Sirin Koca	9 elementer	✉ Delt
Presentasjon	25. september	Sirin Koca	3 elementer	✉ Delt
Testing	20. september	Valerie Smidt	6 elementer	✉ Delt
Observasjoner	... ✉ ★ 9. september	Younes Benhaida	8 elementer	✉ Delt
Minutes-Referat	2. september	Rafey-Ul-Islam Afz	1 elementer	✉ Delt
Data	2. september	Rafey-Ul-Islam Afz	2 elementer	✉ Delt

Figure 30 - Project files on OneDrive

11.12 Project Plan & Management

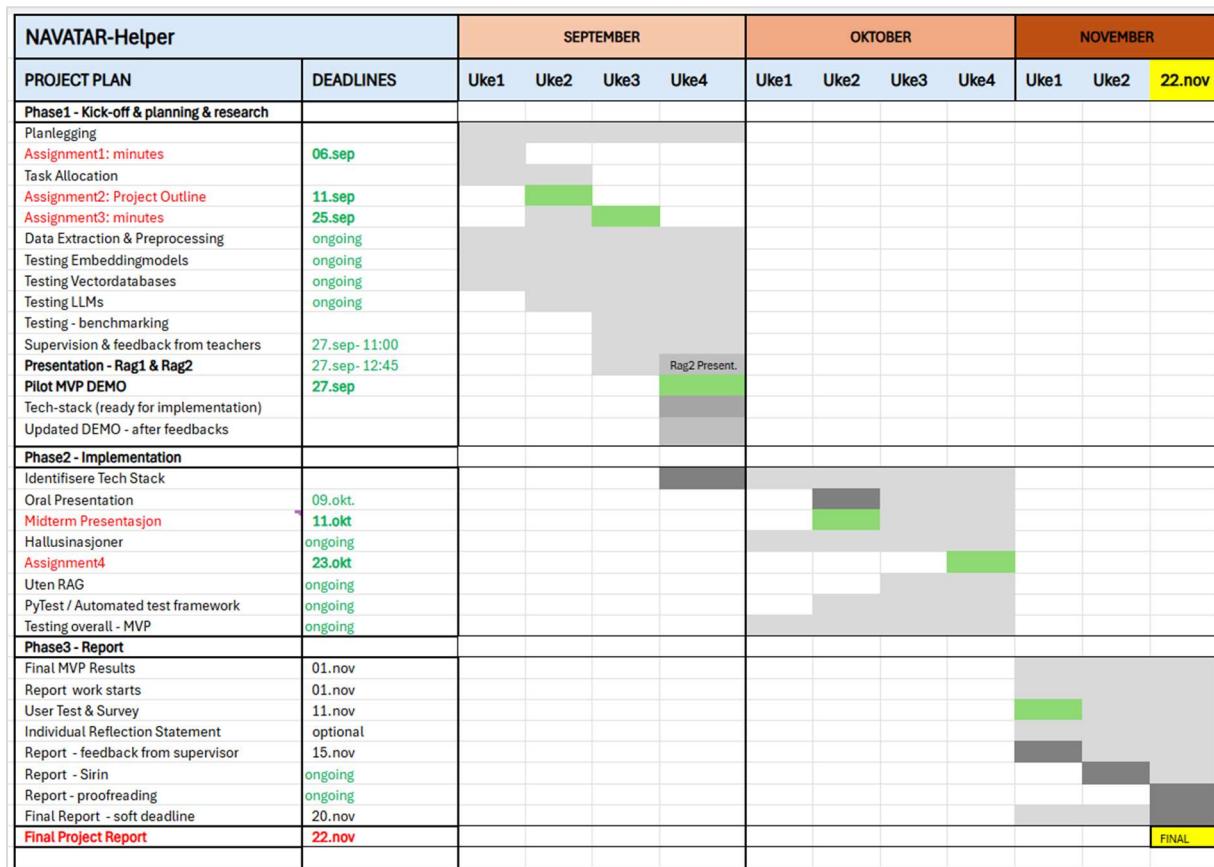


Figure 31 - Tasks and deadlines managed through a Gantt chart we created in Excel

11.13 GitHub Repository & Code Base

GitHub: <https://github.com/sirin-koca/NAVATAR-Helper>

The screenshot shows the GitHub repository page for 'NAVATAR-Helper' (sirin-koca). The repository is public and has 1 branch and 0 tags. The 'Code' tab is selected. The commit history shows several commits from 'sirin-koca' and 'R4f3y'. The repository description is: 'Healthcare chatbot built utilising RAG-framework and LLMs.' It includes links to 'Readme', 'MIT license', and 'Activity'. The 'About' section also lists '0 stars', '2 watching', and '0 forks'. The 'Releases' section indicates 'No releases published' and a link to 'Create a new release'. The 'Packages' section shows 'No packages published' and a link to 'Publish your first package'. The 'Contributors' section lists 'sirin-koca', 'R4f3y', and 'younes2808'. The 'Languages' section shows Python at 100%. Suggested workflows include 'Python package' and 'Pylint'.

Figure 32 - GitHub Repository

11.14 Project Midterm Presentation



NAVATAR Hjelper

1

*Bruk av **Retrieval Augmented Generation – RAG** Rammeverk i Utviklingen av en LLM-basert Helse-Chatbot*

- DATA3750 – Applied AI & Data Science
- Team: Sirin, Younes, Rafey, Valerie, Morten
- Supervisor: Hårek Haugerud, OsloMet

NEXT →

NAVATAR-Hjelper

RAG-basert chatbot som er pålitelig og domeneekspert på NEET (Not in Education, Employment, or Training).

Målet med NAVATAR-Hjelper:

- Domene ekspert - svare på NEET-spørsmål
- Unngå hallusinasjoner (feil eller fabrikkert info)
- Svare "Jeg vet ikke" på spørsmål utenfor NEET
- Svare på språk brukt i spørsmålet (norsk/engelsk)



NEXT →

Figure 33 - Slides from our midterm presentation

11.15 Future of AI

The Future of AI: GraphRAG – A better way to query interlinked documents



vinayakh  MICROSOFT

Nov 13, 2024

GraphRAG is a technique that combines the power of knowledge graphs and large language models (LLMs) to improve the accuracy and relevance of responses to user queries.

All language models are trained on a huge corpus of data. They have some world knowledge and can answer a range of questions about different things. However, due to their probabilistic nature and incomplete world knowledge, especially when it comes to different niches and domains, it's possible to receive incorrect answers. Retrieval Augmented Generation (RAG) helps augment world knowledge with enterprise-specific references, reducing inaccuracies and inconsistencies in the generated text.

Figure 34 - Article published on Microsoft Blog about the GraphRAG technique - combining knowledge graphs and RAG-framework

12. REFERENCES

- Alibaba-NLP. (2024). gte-multilingual-base (current version) [embedding model]. Hugging Face. <https://huggingface.co/Alibaba-NLP/gte-multilingual-base>
- Divya Kumar, K.K. Mishra. (2016). The Impacts of Test Automation on Software's Cost, Quality and Time to Market. Science Direct.
<https://www.sciencedirect.com/science/article/pii/S1877050916001277>
- Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2023). *RAGAS: Automated evaluation of retrieval augmented generation*. arXiv.
<https://arxiv.org/abs/2309.15217>
- LangChain. (n.d). Conceptual Guide. LangChain.
<https://python.langchain.com/docs/concepts/>
- LangChain. (n.d). How to add scores to retriever results. LangChain.
https://python.langchain.com/docs/how_to/add_scores_retriever/
- Milvus. (n.d, 2024). Conduct a Range Search. Milvus.
https://milvus.io/docs/v2.3.x/within_range.md
- Milvus. (n.d, 2024). documentation. Milvus.
<https://milvus.io/api-reference/pymilvus/v2.4.x/About.md>
- MistralAI. (2024). *Mistral-7B-Instruct-v0.1 (current version) [LLM]*. Hugging Face.
<https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1>
- MyScale. (2024, April 17). *Milvus vs Postgres: A Deep Dive into Performance for Vector Data Storage*. MyScale. <https://myscale.com/blog/milvus-vs-postgres-performance-vector-data-storage/>
- NORA – Norwegian AI Research Consortium. (2024, April 16). *Lansering av nye store og åpne norske språkmodeller* [Video]. Youtube.
<https://www.youtube.com/watch?v=tlk161k9Bss&list=PL2YI1C0VZgBYoM2IQcGyNlg0EF-I6CYvH&t=1436s>
- Norallm. (2024). *Instruction-tuned NorMistral-7b-warm (current version) [LLM]*. Hugging Face. <https://huggingface.co/norallm/normistral-7b-warm-instruct>

- Rahmani, H., & Groot, W. (2023). *Risk factors of being a youth not in education, employment or training (NEET): A scoping review*. *International Journal of Educational Research*, 120, 102198.
- Strand, T. (2024, Jan 21). *Deployment of RuterGPT—AI-lab at Ruter—Part 3.2*. Medium. <https://medium.com/ruter-produktutvikling-og-teknologi/deployment-of-rutergpt-ai-lab-at-ruter-part-3-2-08943563bd67>
- Thaker, M. (2023, Nov 14). (Part 1) *Build your own RAG with Mistral-7B and LangChain*. Medium. <https://medium.com/@thakermadhav/build-your-own-rag-with-mistral-7b-and-langchain-97d0c92fa146>
- Vinaykh. (2024, November 13). *The future of AI: GraphRAG – A better way to query interlinked documents*. Microsoft. <https://techcommunity.microsoft.com/blog/aiplatformblog/the-future-of-ai-graphrag-%E2%80%93-a-better-way-to-query-interlinked-documents/4287182>
- Zhao, S., Yang, Y., Wang, Z., He, Z., Qiu, L. K., & Qiu, L. (2024). *Retrieval augmented generation (RAG) and beyond: A comprehensive survey on how to make your LLMs use external data more wisely*. arXiv preprint arXiv:2409.14924. <https://arxiv.org/abs/2409.14924>

Figure List

Figure 1 - Terminology	5
Figure 2 - Main Components of NAVATAR-Helper.....	12
Figure 3- Publicly available local LLMs (ref. section 12, References).....	13
Figure 4 - Milvus vs FAISS	14
Figure 5 - Tentative System Architecture created in September by the project team, OsloMet 2024.	15
Figure 6 - Matrix table showing the tools and technologies evaluated and selected for the project.	16
Figure 7 - System flow – the final app logic, created in November by the project team, OsloMet 2024.	17
Figure 8 - The 100 Question Experimentation.....	19
Figure 9 - Testing NAVATAR-Helper	20
Figure 10 - Testing ChatGPT with various NEET-questions (NonRAG).....	21
Figure 11 - Our customized automated testing process.....	23
Figure 12 - User Test Results	24
Figure 13 - Accuracy comparison of Alibaba, Multilingual-E5, and NO-RAG models at temperature settings of 0.1, 0.2, and 0.3.	25
Figure 14 - Average response times for RAG and NO-RAG across tests, showing RAG's average of 4.13 seconds compared to NO-RAG's 11.03 seconds.	26
Figure 15 - RAG with Mistral-7B and LangChain	30
Figure 16 -How does a multilingual text embedding model work?	31
Figure 17 - Some pictures from development and testing process, setting up PyCharm and GPU-server.....	32
Figure 18 - Backend + Frontend	33
Figure 19 - Frontend - User Interaction	34
Figure 20 - How to evaluate a RAG model?	34
Figure 21 - Manuel testing - test_noRag.py	35
Figure 22 - responses.txt file	35
Figure 23 - Script to generate visualization with PyPlot: 100-Question Experiment	35
Figure 24 - The final technology choice for NAVATAR-Helper	36
Figure 25 - System flow – final version	36
Figure 26 - User Manual	37
Figure 27 - User Survey - Google Forms	38
Figure 28 - Overall results and open questions	39
Figure 29 - Details from user-survey results.....	40
Figure 30 - Project files on OneDrive	41
Figure 31 - Tasks and deadlines managed through a Gantt chart we created in Excel ..	41
Figure 32 - GitHub Repository.....	42
Figure 33 - Slides from our midterm presentation	43

Figure 34 - Article published on Microsoft Blog about the GraphRAG technique - combining knowledge graphs and RAG-framework 44

Wordcount: 4316