KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Technology

Second cycle, 30 credits

Evaluating Techniques for Building AI Assistants in a Specialised Domain

Comparing RAG Techniques and LLMs in Custom Built Assistant for the Canvas LMS at KTH

LUDWIG KRISTOFFERSSON

Stockholm, Sweden, 2024

Evaluating Techniques for Building AI Assistants in a Specialised Domain

Comparing RAG Techniques and LLMs in Custom Built Assistant for the Canvas LMS at KTH

LUDWIG KRISTOFFERSSON

Master's Programme, Computer Science, 120 credits
Date: July 4, 2024

Supervisors: Michael Welle, Fredrik Enoksson
Examiner: Danica Jensfelt
School of Electrical Engineering and Computer Science
Host company: KTH IT
Swedish title: Utvärdering av tekniker för att bygga AI-assistenter i en specialiserad domän
Swedish subtitle: Jämförelse av RAG-tekniker och språkmodeller i en specialbyggd assistent för Canvas på KTH

Abstract

Large Language Models (LLMs) have gained widespread popularity over the past years. Their high level of intelligence and utility is evident. However, integrating them into real-world products and services will require work. This thesis evaluates common tools and technologies used when building applications with LLMs. This includes Retrieval Augmented Generation (RAG), embedding functions and more.

The objective of the research in this thesis is to answer which tools, models and techniques yield the best results. Also, the thesis explores if AI powered applications be built with entirely open source technologies. This is done by building an AI assistant with access to real course rooms in canvas at KTH. The AI assistant is used by real students who are randomly assigned one of the technologies subject to the study. Quantitative and qualitative feedback is collected from the students and analysed in this thesis.

The results show that certain models are preferred by students. Students who got to use GPT-4 by OpenAI reported higher satisfaction with speed, accuracy and usefulness than students who used a smaller and open model developed by Mistral AI.

This thesis concludes that while it is possible to build effective AI powered applications using open source technologies, it is currently easier to create superior experiences with proprietary models. Participants in the study were generally positive to the AI assistant, though some raised privacy concerns. Further research is needed to fully understand the efficacy of various models and techniques in specialised domains, as the one explored in this thesis.

Keywords

AI Assistants, Retrieval Augmented Generation (RAG), Large Language Model (LLM), Information Retrieval (IR), Natural Language Processing (NLP), Vector Embeddings, Learning Management System (LMS)

Sammanfattning

Språkmodeller (LLMs) har blivit mycket populära under de senaste åren. Deras intelligensnivå och användbarhet är tydlig. Men att integrera dem i riktiga produkter och tjänster kräver arbete. Det här examensarbetet utvärderar verktyg och teknologier som ofta används när man bygger applikationer med LLMs. Detta inkluderar Retrieval Augmented Generation (RAG), embedding funktioner med mera.

Syftet med forskningen i det här examensarbetet är att svara på vilka verktyg, modeller och tekniker som ger de bästa resultaten. Dessutom undersöker examensarbetet om AI-drivna applikationer kan byggas med verktyg som uteslutande använder öppna källkod. Detta görs genom att bygga en AI-assistent med tillgång till riktiga kursrum i Canvas på KTH. AI-assistenten används av riktiga studenter som slumpmässigt tilldelas en av de teknologier som studien omfattar. Kvantitativ och kvalitativ feedback samlas in från studenterna och analyseras i detta examensarbete.

Resultaten visar att vissa modeller föredras av studenterna. Studenter som fick använda GPT-4 av OpenAI rapporterade mer positiva svar gällande hastighet, tillförlitlighet och användbarhet än studenter som använde en mindre och öppen modell utvecklad av Mistral AI.

Examensarbetet drar slutsatsen att medan det är möjligt att bygga effektiva AI-drivna applikationer med öppna modeller och tekniker, är det för närvarande enklare att skapa smartare system med proprietära modeller. Deltagarna i studien var generellt positiva till AI-assistenten som utvecklades för studien, även om vissa uttryckte oro över integritetsfrågor. Ytterligare forskning behövs för att fullt ut förstå effektiviteten hos olika modeller och tekniker inom specialiserade domäner, som den som utforskas i detta arbete.

Nyckelord

AI-assistenter, Retrieval Augmented Generation (RAG), Stora språkmodeller (LLM), Informationssökning (IR), Naturlig språkbehandling (NLP), Vektorinbäddningar, Lärplattform (LMS)

Acknowledgments

I would first like to thank my two supervisors on this thesis. First Fredrik Enoksson from the Unit of Digital Learning at KTH, for helping me conceptualise and build an assistant that could actually work in the digital environment at KTH. Additionally, helping me with getting in touch with several examiners who participated in the study, helping me with the logistics of getting access to the course room, and much more. I will miss our weekly catch-up that we've had throughout the project.

Secondly, I would like to thank my supervisor on the research in the thesis, Michael Welle from the Division of Robotics, Perception and Learning. Michael has been a great support helping figure out what research I should do within my project. I appreciate that you've kept me on track throughout the project, without your guidance I'm sure I would've still been building on those early prototypes.

I would like to thank all the participating teachers and TAs. Without the courses who voluntarily allowed the assistant to be deployed in their course rooms this research could not have been completed. These include Marcus Lithander, Malin Jansson, Dena Hussain, Christian Smith, Richard James Glassey, Patric Jensfelt. An extra big thanks to Antonio Maffei and Fabio Marco Monetti who not only participated but also helped in constructing the study.

Without the funding provided from KTH Innovation the research in this thesis would not have been possible. The grants they provided me last year for kthGPT helped me with that and this thesis. Thanks to Hannes Eder Öhrström for all advice and setting me up with the funding last year.

I would also like to thank Sebastian Ware for inspiring me to pursue this subject for my Master thesis. I've learned a lot about LLMs and RAG. I'm not sure I would've pursued this thesis unless you advised me to do so, thanks!

Additionally, I would like to thank Rebecca Stower for helping me design the feedback questions and how they should be integrated into the software. Without your input I would not have known how to do that. Her advice prevented me from pursuing a potentially misguided path at a crucial time in the master's thesis, which could have led to significant hurdles down the line.

The support I've had from my friends through all of my five years at KTH have been paramount. I'm confident that I wouldn't have made it without them. Even though we've been on our own with the master thesis, and many courses before that, the daily support and camaraderie have made these years some of the best years in my life.

Lastly, I would like to thank my employer, Stockholm Code Group, which has supported me in pursuing my degree. They hired me 6 months before enrolling at KTH, knowing that I would become a fulltime student, not many firms would do so. I've spent countless hours studying in our offices, and that office, along with the people in it, have been just as significant in helping me finish this degree. Thanks for supporting my side-hustle as a full time student. A special thanks to Patrick Zeits, who convinced me to pursue an education in the first place. I'm grateful you showed me *why* higher education was useful.

Stockholm, July 2024
Ludwig Kristoffersson

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

BERT	Bidirectional Encoder Representations from Transformers
CBOW	Continuous Bag-of-Words
CNN	Convolutional Neural Networks
EC2	Amazon Elastic Compute Cloud
ECM	Expectation-Confirmation Model
ECR	Amazon Elastic Container Registry
ECS	Amazon Elastic Container Service
GAN	Generative Adversarial Network
GPT	Generative Pre-trained Transformers
GQA	Grouped-Query Attention
GRU	Gated Recurrent Units
GSM8K	Grade School Math 8K
GUI	Graphical user interface
IR	Information Retrieval
LLM	Large Language Models
LMS	Learning Manegement System
LSTM	Long Short-Term Memory
MBPP	Mostly Basic Programming Problems
MMLU	Massive Multitask Language Understanding
MTEB	Massive Text Embedding Benchmark
NLP	Natural Language Processing
PIQA	Physical Interaction: Question Answering
POC	Proof-of-concept
RAG	Retrieval Augmented Generation
RDS	Amazon Relational Database Service
REST	Representational State Transfer
RNN	Recurrent Neural Network

S3	Amazon Simple Storage Service
SDG	Sustainable Development Goal
seq2seq	Sequence-to-sequence
SMoE	Sparse Mixture of Experts
SNS	Amazon Simple Notification Service
SWA	Sliding window attention
TAM	Technology Acceptance Model
TF-IDF	Term Frequency-Inverse Document Frequency
URL	Uniform Resource Locator

Chapter 1

Introduction

1.1 Background

This degree project will investigate Large Language Models (LLM) and Retrieval Augmented Generation (RAG) systems in the form of deploying an AI-assistant in canvas course rooms. The degree project will investigate how to evaluate these systems in very specialised domains and benchmark various models, approaches and techniques.

The reason this research is important is that LLMs have gained widespread attention and we are likely to see large-scale adoption of these models into various applications. Understanding how to benchmark and evaluate these systems in specialised domains will be crucial to understand how to build these systems, which techniques to use, and which models work well. One such technique is RAG, which is a way of providing specialised knowledge to a model. Understanding how to do that well is crucial for understanding how to build an LLM-based application well.

Many organisations need to, due to commercial and regulatory compliance, host all AI-models themselves. This aspect is also interesting to evaluate, i.e. how well open source and commercially licensed models compare against the closed source models, such as GPT-4 by OpenAI.

The research will be carried out within the e-learning management object at KTH, who are responsible for the digital learning environment at KTH. The object consists of two teams at the KTH IT department and one team at the digital learning unit at the ITM-school. The university hosts thousands of courses with domain specific information, such as assignments, lectures and schedules, that aren't part of the public domain and therefore not part of the training set of LLMs.

All the work done by KTH IT aims to improve the operations at the university. Among this is reducing the administrative burden undertaken by teachers and teaching assistants (TAs). KTH IT wants to investigate if AI-assistants can be deployed into the canvas course rooms to reduce the workload of teachers and TAs which would help them focus on teaching, helping students and improve the quality of the education. KTH IT wants to see if it's feasible to deploy an AI assistant into the canvas course rooms.

1.2 Problem

LLMs have gained widespread use since its popularisation by ChatGPT. Their abilities to summarise large bodies of text and follow user instructions have proven very useful in many contexts. However, considering their limited context window (and drawbacks of models with larger context window [1]) deploying useful applications with a chat based interface still rely upon integrating a RAG system, introduced by Lewis et al. [2]. These can retrieve relevant information needed to answer a user's query from outside data sources and inject them into the conversation.

Some announced but currently unreleased models, such as the gemini family of models [3], have been reported to show great recall performance and reasoning abilities over millions of tokens. This could significantly reduce the importance of RAG systems in applications which utilise LLMs and external datasets to create intelligent systems with domain specific knowledge. However, even though no exact figures are presented by the Gemini team, inference speed (the time taken to produce a response to a prompt) seems to be significantly slower than with shorter contexts. This would again highlight the importance of efficient RAG systems. Still, other approaches than traditional GPUs have been shown recently [4] by the Groq team to greatly increase inference speed.

Evaluating large language models is notoriously difficult. There are objective and automated metrics that can be used for tasks such as evaluating a model's summarisation capabilities, as shown by Basyal and Sanghvi [5]. However, for more complicated evaluations it gets trickier. In their seminal instructGPT paper Ouyang et al. [6] at OpenAI try to evaluate *"how well a model can follow instructions"* which is a very subjective question. They essentially relied upon human labellers to judge the overall quality of each response generated by the model.

In their Gemini-paper the Gemini team discuss the benchmarks used for their largest model. The team states that benchmarks are often designed to test shorter prompts whereas their longer prompts challenge tests used in traditional evaluation methods that rely heavily on manual evaluation. This highlights the relevance of good evaluation metrics. Regardless of context size or inference speed, evaluation of models tends to be very general. Which makes sense, when considering their general application.

When releasing their Mixtral model [7] the Mistral AI team used a range of benchmark tests, such as Massive Multitask Language Understanding (MMLU), Physical Interaction: Question Answering (PIQA), Grade School Math 8K (GSM8K) etc. MMLU [8] benchmarks a LLMs proficiency in understanding and reasoning across various subjects such as humanities, STEM, and professional and everyday knowledge, by evaluating its performance on 57 tasks, to test its ability to generalise and apply knowledge. PIQA [9], evaluates a language model's understanding of physical commonsense by asking them to predict the outcome of physical interactions in various scenarios through multiple-choice questions. GSM8K [10] tests the ability to solve elementary-level mathematics word problems.

Evaluation of how well LLMs perform is an open research question. As shown above LLM developers often utilise multiple testsuites. These are oftentimes, very general tests. When implementing LLMs in practical applications good performance often relies upon very good raw summarisation performance and reasoning abilities. Since the domain specific knowledge is provided to the model, raw built-in knowledge isn't crucial. It is more important for the model to learn the task at hand using very few examples and within the given domain understand the question being asked by a user. Further, as argued by by Siriwardhana et al., the training data of LLMs include the knowledge of datasets such as Wikipedia [11] which means that evaluation methods in very specialised domains hold higher value than generalised domains. These brand new domains, that with certainty haven't been seen during training, tests the models zero-shot, and depending on the implementation, few-shot learning abilities.

1.2.1 Research question

The research question for this project is *Which language model and which retrieval techniques do students prefer using?* and *Is it possible to deploy an AI-assistant using a completely open source toolchain?*.

I believe the answer to the first question is that the closed source alternatives will be preferred by the students, however, I think the results will show it is possible to deploy an open source based AI assistant too.

1.2.2 Original problem and definition

The core challenge addressed in this thesis is the effective deployment and evaluation of AI-assistants powered by LLM and RAG techniques in a specialised domain, specifically within the Learning Manegement System (LMS) of Canvas course rooms at KTH. This involves assessing the practicality and efficiency of integrating AI-Assistants built upon LLMs and RAG techniques into the educational settings to aid in reducing administrative burdens on educators and enhancing student interaction with course materials.

The original problem stems from the need to understand whether AI-assistants can effectively handle the domain-specific data intrinsic to educational platforms that are not included in their initial training datasets. Furthermore, the project aims to compare the efficacy and acceptability of open-source versus proprietary AI models in real-world educational applications.

1.3 Purpose

The purpose of this thesis is two-fold: firstly, to innovate within the educational technology space by integrating AI-assistants to potentially reduce workload and improve informational access within Canvas course rooms. Secondly, the thesis aims to contribute to academic knowledge by providing empirical data on the performance of these AI systems in a controlled educational setting. The dual purpose of this thesis ensures it not only investigates the immediate needs of KTH's digital learning environment but also enriches the scientific community's understanding of applied AI within a specialised domain, such as education.

This research is intended to benefit educational institutions by potentially offering a tool that improves operational efficiency and students by providing an alternative, possibly more effective way of interacting with course content. In addition the research will bring benefits for researchers within AI and education. Ethically, the study focuses on the sustainable development of AI technologies by emphasising open-source solutions, aiming to democratise advanced technological developments and reduce reliance on proprietary models.

1.4 Goals

1. **Technological Efficacy:** To evaluate the accuracy, speed, and reliability of responses by AI-assistants utilising both proprietary and open-source models in handling domain-specific content, such as the course rooms in canvas.

2. **User Preference:** To understand the preferences of students regarding the usability, information quality, and overall experience of interacting with an AI-assistant built upon different models and retrieval techniques.

3. **Operational Feasibility:** To assess the feasibility of integrating an AI-assistant built on fully open-source technologies within an academic setting, considering logistical, technical, and regulatory constraints.

4. **Educational Impact:** To explore the potential of AI-assistants to reduce administrative burdens on educators and improve information accessibility for students.

5. **Comparative Analysis:** To perform a comparative study between various LLM models and RAG-techniques.

1.5 Research Methodology

This project primarily employs an empirical study based on data collection and quantitative analysis of responses collected from students using the software designed for this study. There are some forms distributed that will be used to collect insights using a qualitative approach. Both of these will be used to evaluate the implementation of AI-assistants in the educational domain.

1.5.1 Experimental Design and Implementation

The following section will outline the considerations the experiments in this thesis must consider. This includes both requirements for the software design and architecture, and the questions to distribute in the course.

Model selection Different models, including proprietary and open-source, with different sizes (number of active parameters), will be tested. The relevant models will be included in the study for testing.

RAG technique selection Various configurations of RAG systems will be tested to identify the most effective method for enhancing the AI's responses with respect to the layout of the data in Canvas course rooms. The relevant techniques will be included in the study for testing.

Implement AI Assistant Design and implement the system that will be used in the study. This includes application logic, user interface design, etc.

Construct study questions Craft the questions that will be asked to students and implement them in the AI assistant.

Courses to include in the study Find willing course administrators that want to participate in the study with their students.

1.5.2 Evaluation Design

The evaluation of the experiments is key to ensure the study is methodologically sound. The experimental setup is described in detail in chapter 3. On a high-level the experiments must consider the following to ensure its setup is robust and meaningful.

Study Participants The study will involve students using the AI-assistant and ask them to provide feedback on their experiences. This is a good selection of participants, considering the goals of the research.

Experimental Setup Controlled experiments will be conducted where participants use different configurations of the AI-assistant for typical student questions. These configurations are randomly assigned.

Data Collection Methods Data will be collected through integrated survey questions within the chat interface, capturing real-time feedback on the AI-assistant's performance and student satisfaction.

1.5.3 Analysis Techniques

The analysis techniques are similarly described in greater detail in chapter 3. However, briefly its method can be broken down into two categories. These are;

Quantitative Analysis The data collected from the systems performance, and the responses collected through the multiple choice questions, asked to the participating students, will be collated into tables and charts. Using statistical methods this thesis will present results on which models, tools and techniques lead to the highest student satisfaction.

Qualitative Analysis Feedback and open-ended responses will be analysed textually to understand user perceptions and contextual effectiveness of the AI-assistant.

This methodology was chosen for its ability to provide a comprehensive evaluation of both the technical capabilities and the practical usability of AI-assistants, offering insights into their potential benefits and limitations in the specific context for this study.

1.6 Delimitations

This project has several delimitations that define the scope and boundaries of the research to ensure a focused and manageable study. The key delimitations are;

- **Model Scope:** The project will not involve the development of new models or the fine-tuning of existing models. This includes LLM and embedding functions. The study will utilise pre-trained models offered by bigger vendors or the open source community.
- **Data Limitations:** Only existing courses within KTH's Canvas LMS will be utilised for the study. No new course content will be created, and no modifications will be made to existing course materials beyond what is necessary for the integration and testing of the AI-assistants.
- **Course Data Access:** The project will not use Canvas APIs for data integration. All interactions with the Canvas platform will be through existing interfaces, or data will be scraped and used from the Canvas web interface.
- **Geographic and Cultural Constraints:** The study is limited to the KTH environment, which may not represent other educational settings in different cultural or geographic contexts or languages. The findings might not be directly transferable to other institutions or countries without additional localisation and adaptation.

Chapter 2

Background

This chapter provides the necessary background for understanding the research conducted within this thesis. This chapter also showcases the related work for this thesis and how the research relates to it.

2.1 Neural Networks

Neural network models are a type of models within the broader field of machine learning whose design have been inspired by human brains. These models allow computers to recognise patterns and solve complex problems. The backpropagation algorithm was popularised by Rumelhart, Hinton, and Williams [12]. This algorithm efficiently computes the gradient of the loss function with respect to the weights of the network by propagating the error back from the output layer to the input layer. This method is critical to understand all machine learning pipelines because it enables the network to adjust its weights in a way that minimises the error, thereby improving the model's predictions over time.

Building on backpropagation, Yann LeCun et al. [13] introduced the Convolutional Neural Networks (CNN) architecture in 1998. These are a specialised kind of neural network for processing data, such as images, which can be converted to a matrix. CNNs utilise layers with convolving filters that apply the learned weights across subsections of the input data. This reduces the amount of parameters in the network and improves its efficiency.

These are two steps in the evolution of neural network models, particularly the developments in CNNs and other deep learning technologies, are central for setting the stage for even more complex architectures aimed at processing not just visual data, but sequential data such as text. This will eventually lead to Large Language Models (LLM), which leverage deep learning techniques to understand and *generate* human language. LLMs are built upon the principles of neural networks. Understanding the models we commonly refer to as LLMs involves understanding models such as Transformer models, Bidirectional Encoder Representations from Transformers (BERT), and other encoder-decoder networks.

2.1.1 Recurrent Neural Networks (RNNs)

A Recurrent Neural Network (RNN) is a type of neural network that is good for modelling sequential data. They are significantly different from other neural networks in their ability to maintain memory of previous inputs using an internal state. This state which is maintained inside the network while it's running, will influence the network's output. RNNs proved to be fundamental in tasks where context was crucial, such as language modelling and generation of text.

In an RNN, each neuron, its most basic building block, processes a part of the sequence, receiving both the current input x_t and the output from the previous step h_{t-1}, this is known as the "hidden state". The core of an RNN operation involves updating this hidden state using:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b)$$

where W_{hh} and W_{xh} are the weights for the hidden state and input, respectively, and b is a bias. The updated state h_t is used in the next step to generate the output y_t via:

$$y_t = W_{hy}h_t + b_y$$

However, RNNs often struggle with maintaining a longer context due to problems like vanishing and exploding gradients, as written by Hochreiter and Schmidhuber [14]. This was a problem other RNN models tried to mitigate as it significantly reduce their usefulness in various tasks. The vanishing gradient problem makes it difficult for the RNN to learn connections between events that occur at longer distances in the input sequence because the gradient of the loss function decays exponentially with the length of the input sequence.

This led to the development of more sophisticated variants like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU)s were developed. LSTMs [14], use input, output, and "forget gates" to manage information flow, which allows them to maintain stable gradients. GRUs, which was proposed by Cho et al. [15], simplifies this by merging the gates and states, reducing complexity while preserving performance across various tasks.

2.1.2 Sequence-to-Sequence Models

Sequence-to-sequence (seq2seq) models are designed to process sequences of data, such as text or speech, and generate corresponding output sequences. Sutskever et al. [16] were the first to introduce these models which typically consist of two main components: an encoder and a decoder. The encoder will process the input and convert it into a dense vector. This vector encodes the entire input sequence which is then passed to the decoder, which generates the output. This architecture proved very useful in certain tasks such as translating text between languages. Bahdanau, Cho, and Bengio built upon this concept with attention mechanisms [17] which would allow the decoder to focus on a specific piece of the input for small parts of the output, which improved the models ability to focus on longer sequences.

2.1.3 Transformer Models

The Transformer model, introduced by Vaswani et al. [18], was a new approach for seq2seq networks, with a self-attention mechanism which was different from the recurrent design of RNNs. The new transformer architecture introduced by Vaswani et al. allowed the network to weigh the importance of different tokens in the input data irrespective of their sequential position. Where a token is a sequence of characters that can be treated as a single logical entity in the input and output sequence.

The key innovation of the Transformer is its ability to handle dependencies between single tokens or sequences of tokens at long distances from each other. This makes the transformer architecture especially good at understanding context in text data.

The introduction of the transformer model was foundational in the field, and today most models use this architecture, see section 2.3 and 2.3.2.

2.1.4 BERT and Advances in Encoder-Decoder Models

BERT was introduced by Devlin et al. [19] in 2018 and was a major improvement within natural language processing. The BERT model optimised token representations bidirectionally which means that it was refining the understanding of each token by looking at the tokens before and after each token. BERT was built on the transformer model's encoder which allowed for pre-training on large text corpora, followed by fine-tuning for various tasks such as sentiment analysis and question answering.

Encoder-decoder models are important in machine learning for tasks that involve converting one sequence into another, such as machine translation or speech-to-text. In this type of model the encoder processes the input sequence and compresses information into what's known as a context vector, this is a condensed representation of the input data. The decoder takes this context vector and generates an output sequence token by token. Each of these two components may be built using recurrent networks, convolutional networks, or more commonly nowadays, transformer architectures.

In contrast to traditional encoder-decoder models, encoder-only models, such as BERT, focus on generating an output based on an input without the need for a decoder. These models are typically used for tasks that require deep understanding of language context like sentence classification.

Decoder-only models, like the Generative Pre-trained Transformers (GPT) (see section 2.3), focus on generating sequences from a given context or starting point. These models are very good in situations where the model needs to exhibit "creative" properties, such as when generating text completions.

Parallel to BERT, other encoder-decoder models like the Transformer [18] and seq2seq networks with attention mechanisms [17] have shown great results when translating sequences in tasks like machine translation, exemplified by Google's Neural Machine Translation system [20], and speech recognition, as seen in Apple's Siri voice assistant [21].

2.2 Generative AI

Generative AI is a term used to describe a subset of artificial intelligence technologies that are designed to create new content. This can be images such as with DALL-E [22], text with models like GPT-3 [23] or movies [24]. These models are capable of generating realistic and arguably novel outputs by understanding and simulating the underlying structure of the training data. One of the most popular frameworks in Generative AI includes Generative Adversarial Network (GAN)s, introduced by Goodfellow et al. [25], which consist of two neural networks, the generator and the discriminator. These two networks will compete against each other. The generator creates items that are as realistic as possible, and the discriminator evaluates them. This process runs until the discriminator can no longer accurately separate generated items from the training data.

2.3 State-of-the-Art Large Language Models

LLM represent a significant breakthrough in Natural Language Processing (NLP). They are capable of understanding and generating text similar to that written by humans. In recent years, several cutting-edge LLMs have been developed by prominent companies and research institutions that have gained wide-spread use. This section gives an overview of some notable examples of these advanced LLMs.

2.3.1 OpenAI's GPT Series

OpenAI's GPT series of language models have over the past few years featured some of the most widely used language models. GPT-1 was first released in 2017 followed by GPT-2, GPT-3, and GPT-4 (with various variants of these models). GPT-3, in particular, with its 175 billion parameters, has demonstrated strong capabilities in tasks such as text completion, question answering, and even code generation [23]. These models are some of the most widely used models, primarily due to their popularisation by the product from the same company, ChatGPT *.

2.3.2 Mistral

Mistral is a French firm that has released a few models that have gained widespread adoption in the open source community. As of writing, *Mistral-7B-Instruct-v0.2* had 2,297,845 million downloads on huggingface last month †, and Mixtral-8x7B-Instruct-v0.1 had 628,927 ‡.

Mistral 7B v0.1 [26] was their first major model to get widespread notoriety. The model is a 7-billion-parameter language model which was small enough to run on consumer-grade GPUs. The model utilised Grouped-Query Attention (GQA)[27] and Sliding window attention (SWA) [28] techniques to achieve impressive results across various benchmarks, including reasoning, mathematics, and code generation tasks. *Mistral 7B v0.1 instruct* is a related fine-tuned model.

The "instruct" version of generative AI models, such as the Mistral 7B, has been fine-tuned to follow prompted instructions. In contrast, the base model simply generates output based on the provided prompt. This process

*chat.openai.com (accessed on July 4, 2024)

†The huggingface page for Mistral-7B-Instruct-v0.2 (accessed on June 6, 2024)

‡The huggingface page for Mixtral-8x7B-Instruct-v0.1 (accessed on June 6, 2024)

was first published by the team at OpenAI [6], however it's also employed by Mistral and other model vendors. This approach is commonly used for models deployed in AI assistants or chat applications.

The *Mixtral of Experts* model [7], is a variant of the Mistral model that introduces a Sparse Mixture of Experts (SMoE) architecture, as described by Jiang et al. *Mixtral-8x7B-Instruct-v0.1* employs 8 feedforward blocks (experts) in each layer, with a router network selecting two experts for processing and combining their outputs at each timestep. The model has access to 47 billion parameters, but effectively only utilises 13 billion parameters during inference, which makes the model easier to deploy on GPUs with less amounts of memory.

2.3.3 Google's Language Models

Google has two major families of model, the first being the Gemini family, as introduced in a series of papers by Google's team [29], consists of models like Gemini Ultra, Pro, and Nano, each of these models are designed for specific applications and more importantly size of GPU. Where the larger models require enterprise-grade GPUs that are expensive to operate. Gemini 1.5 extended on these models with an even larger context window by effectively processing and recalling information across millions of tokens in a multi-modal context (tokens include both text, audio and image tokens) [3]. This is the first model to demonstrate resilience to the problem first described by Nelson et al. where the model would be biassed towards instructions or data in the beginning and end of larger prompts [1].

Goggles Gemma family of models [30] represents Google's effort to provide state-of-the-art, lightweight models to the open source community. These models, available in sizes of 2 billion and 7 billion parameters. The models demonstrate worse performance against their Gemini class of models across all tasks such language understanding and reasoning. However, the Gemma models' size makes them easier to deploy on smaller consumer-grade GPUs.

2.3.4 The LLama family of models

In February 2023, Meta AI released LLaMA [31] in four distinct sizes: 7, 13, 33, and 65 billion parameters. The model utilised features such as SwiGLU activation functions, rotary positional embeddings, and root-mean-squared layer-normalisation to achieve comparable results to OpenAIs GPT-

3 model. Despite being initially released under a noncommercial licence, the weights of LLaMA were leaked, prompting widespread unauthorised use. This accelerated its adoption across various applications.

Later in July of 2023, Meta released LLaMA-2 [32] which was built upon the foundational models of its predecessor with enhanced data sets of 2 trillion tokens, fine-tuning capabilities, and improved dialogue system performance through specialised LLaMA-2 Chat models, these are similar to the instruct models mentioned in section 2.3.2. LLaMA-2 had a 40% larger training corpus and extended the context length to 4,000 tokens. The release included model sizes from 7 to 70 billion parameters. These models were released under a similar licence to the first LLaMA models.

Recently, in April 2024, Meta AI released LLaMA-3, this time with two models, one 8 billion parameter model and one 70 billion parameter model. These were open source and available online * from day one under a commercial licence. The model was pre-trained on approximately 15 trillion tokens. Meta announced an, as of writing, future release of a 400 billion parameter model.

2.3.5 Notable other vendors

Besides the major players such as OpenAI, Google, and Meta, there exists a vast array of players, of varying size, that also develops language models. These include, but are not limited to, Anthropic, IBM and DeepMind (which is also a part of Google).

2.4 Prompt engineering

Prompt engineering is the name given to the technique that evolved from the use of language models. This is the task of optimising the performance of a LLM such as GPT-4, LLaMA, and others. This involves crafting the input text, or *"prompt"* to these models in a way that guides them to produce desired outputs [33, 34].

Prompt engineering is defined as the practice of designing input prompts that maximise the efficacy and accuracy of LLM outputs. It is a key factor in the success of deploying LLM-based applications. The process of prompt engineering involves several key techniques. A prompt should, according to Chen et al. include clear instructions and enough contextual details to guide

*The GitHub repository for LLaMA-3 (accessed on July 4, 2024) the model towards providing the expected answer in the expected format. There are numerous advanced techniques such as "role-prompting", zero-shot, one-shot, and few-shot prompting that can improve the performance of LLM.

For instance, Kathiriya et al. [33] demonstrates that role-prompting produces responses with heightened professional relevance. Similarly, Chen et al. highlight how few-shot prompting can refine the model's ability to perform complex analytical tasks by providing some targeted examples. Both of these studies show how prompt engineering techniques can improve performance.

Figure 2.1, taken from the paper published by Chen et al. [34] illustrates an example of role-prompting. In this example the LLM is instructed to assume the role of an expert in artificial intelligence, which aligns its responses with specific professional knowledge.

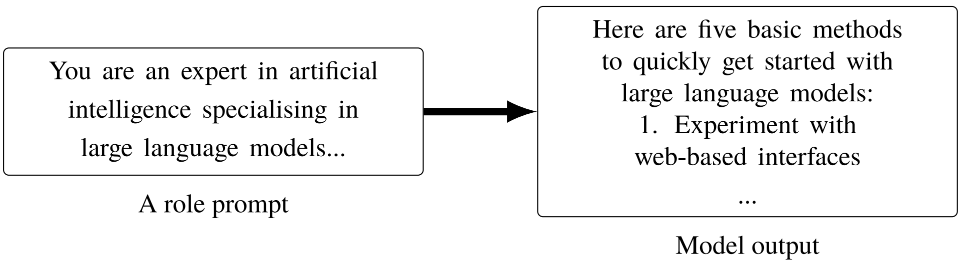

Figure 2.1: Role prompting example.

Another technique known as few-shot prompting, is shown in figure 2.2, taken from the paper written by Kathiriya et al. [33]. With this technique the model is provided with multiple examples to better understand the task. If only one example is given, this is referred to as "one-shot" prompting. Similarly, if no example is given, then the prompt is referred to as a "zero-shot" prompt.

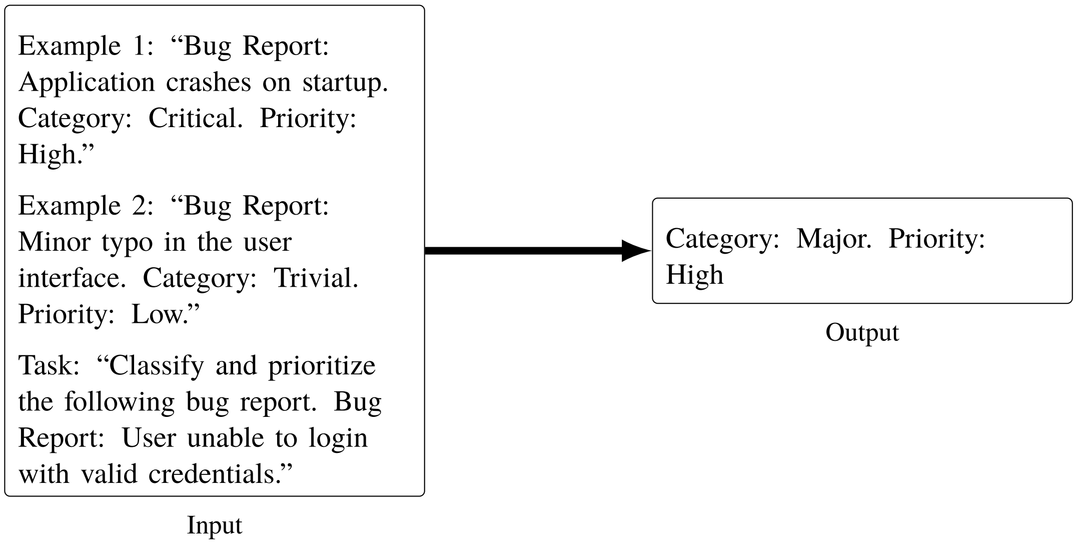

Figure 2.2: Few-shot prompting example.

2.5 Evaluating LLM performance

When an LLM vendor, such as Mistral, releases a new model its performance is evaluated using a series of well-established benchmarks. These benchmarks are essential for understanding the model's capabilities in various cognitive tasks. This includes tasks such as mathematical reasoning, language understanding and program synthesis. This practice helps to quantify the models' performance and provides a method to compare its performance against previously released models [10, 35, 36].

When Mistral released its first major model they used Table 2.1 to compare its results against the LLaMA family of models. There are numerous benchmarks included in the table that test various abilities of the model. For example, the GSM8K benchmark includes thousands of grade-school level maths problems that are designed to test mathematical reasoning [10]. Benchmarks like Mostly Basic Programming Problems (MBPP) assess a model's ability to understand and generate programming code from natural language descriptions [35]. A test like MMLU measures general world knowledge and problem solving ability [8]. Lastly, the PIQA benchmark, challenges a model with physical common sense questions [9]

Model	MMLU	HellAs	PIQA	Arc-e	Arc-c	HumanE	MBPP	Math	GSMBK
LLaMA 2 7B	44.4%	71.1%	77.9%	68.7%	43.2%	11.6%	26.1%	3.9%	16.0%
LLaMA 2 13B	55.6%	70.7%	80.8%	75.2%	48.8%	18.9%	35.4%	6.0%	34.3%
LLaMA 1 33B	56.8%	83.7%	82.2%	79.6%	54.4%	25.0%	40.9%	8.4%	44.1%
LLaMA 2 70B	69.9%	85.4%	82.6%	79.9%	56.5%	29.3%	49.8%	13.8%	69.6%
Mistral 7B	62.5%	81.0%	82.2%	80.5%	54.9%	26.2%	50.2%	12.7%	50.0%
Mixtral 8x7B	**70.6%**	**84.4%**	**83.6%**	**83.1%**	**59.7%**	**40.2%**	**60.7%**	**28.4%**	**74.4%**

Table 2.1: Table used by Mistral to compare the performance of *Mixtral 8x7B* to *Mistral 7B* and the Llama family of models.

All of these benchmarks are essential for developers and researchers to understand the limitations and capabilities of AI models. They ensure continuous improvements and innovations in the field. Each benchmark is sourced and created differently. Some models are tested against just a single benchmark such as OpenAI's codex model (now deprecated) [36]. However, most general purpose language models such as those released by Mistral, OpenAI, Meta, Google and more use a common set of benchmarks such as those in table 2.1 [7, 37, 31, 3].

2.6 Web crawling

Web crawling is a technique to systematically browse the World Wide Web to index the content of websites for search engines and other applications using automated programs known as web crawlers [38, 39]. This is a process that's crucial for the operation of search engines.

A web crawler starts with a list of URLs to visit. As the crawler visits these URLs, it identifies all the hyperlinks on the page and adds them to a database of known URLs to visit. After visiting a URL the crawler employs a method of selecting the next url to visit, which may be one of the hyperlinks it just found on the current page, or any url it might have found before. This method may vary depending on the implementation of the crawler. This process continues until a defined stop condition.

While the primary application of web crawling is in web search engines, it can also be used within various other domains. Web crawlers can be used for everything from monitoring changes in web pages to gather data from specific intranets or corporate knowledge bases.

Implementing an efficient web crawler involves addressing multiple technical challenges. These are primarily constructing an efficient crawler that can visit and process urls at scale. Additionally, the crawler must be able to index the content found on those websites, which may include various media types such as plaintext, images or document formats such as PDF.

2.7 Information Retrieval

Information Retrieval (IR) refers to the process of returning relevant information from a corpus of documents. The field primarily focuses on the retrieval of text data and is a core part of many applications such as search engines or AI agents.

The objective of information retrieval is to find material within an unstructured database [40]. This usually involves resolving the relevant documents in response to a user query. Information retrieval systems are usually measured against precision and recall metrics. These show how relevant the documents returned were, and how many of the relevant documents were returned.

$$\text{Precision} = \frac{\text{Number of Relevant Documents Retrieved}}{\text{Total Number of Documents Retrieved}} \tag{2.1}$$

$$\text{Recall} = \frac{\text{Number of Relevant Documents Retrieved}}{\text{Total Number of Relevant Documents in the Corpus}} \tag{2.2}$$

The core of IR is indexing and search algorithms. To index a corpus means processing all the documents in the corpus into a data structure that can later be used for retrieving docs. Search algorithms utilise this index to find documents that match the user's query [40].

The second problem of IR is to rank the returned documents. This is a problem with many possible solutions.

2.7.1 Term frequency inverse document frequency

Term Frequency-Inverse Document Frequency (TF-IDF) is a measurement used to evaluate how important a word is to a document in a collection or corpus. This means it is a relative metric that is unique to the corpus being indexed. TF-IDF is calculated by multiplying two values

1. How many times a term appears in a document

2. The inverse document frequency of the term across a set of documents

Term frequency is calculated using the following formula;

$$\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \tag{2.3}$$

The inverse document frequency is calculated using;

$$\text{IDF}(t, D) = \log\left(\frac{\text{Total number of documents in the corpus } D}{\text{Number of documents containing term } t}\right) \tag{2.4}$$

The complete formula for TF-IDF becomes;

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D) \tag{2.5}$$

This formula means the relevance for a token increases with the number of times that term appears in the document, but is offset by the frequency of the term in the entire corpus. This is a good way of adjusting for the fact that some words are generally more common than others, such as "a", "the", etc. [40].

2.7.2 Embedding Functions

Vector embeddings are a way of representing text or other media content, such as images, as a numerical vector that encapsulates their features. Figure 2.3 illustrates how a token, in this case *cat* and *dog*, is encoded into a vector.

$$\texttt{cat} \rightarrow \{0.042, 0.112, 0.236, 0.368, 0.491, 0.623, 0.784, 0.895, \ldots, 0.931\}$$

$$\texttt{dog} \rightarrow \{0.157, 0.209, 0.330, 0.501, 0.579, 0.619, 0.755, 0.832, \ldots, 0.874\}$$

Figure 2.3: Example embeddings for "cat" and "dog" strings.

Vector embeddings "embed" the features of their input. For text content such a feature could be something abstract about a word that's even true in several languages. In text processing, one typically leverages the neural network of a language model to understand the contexts and co-occurrences of tokens. These networks have usually been trained on very large corpora of text and are thereby very good at placing semantically similar tokens close to each other in a vector space. For example, *football* and *soccer* may appear in similar contexts, leading the network to locate them near each other in a "meaning space", as can be seen in figure 2.4. Measured with something like levenshtein distance, the words are very far from each other, even though we know they are synonymous in many contexts. Processing text with a neural network and representing it with a vector can allow computers to perform complex tasks like text prediction with an understanding akin to human cognitive judgments [41].

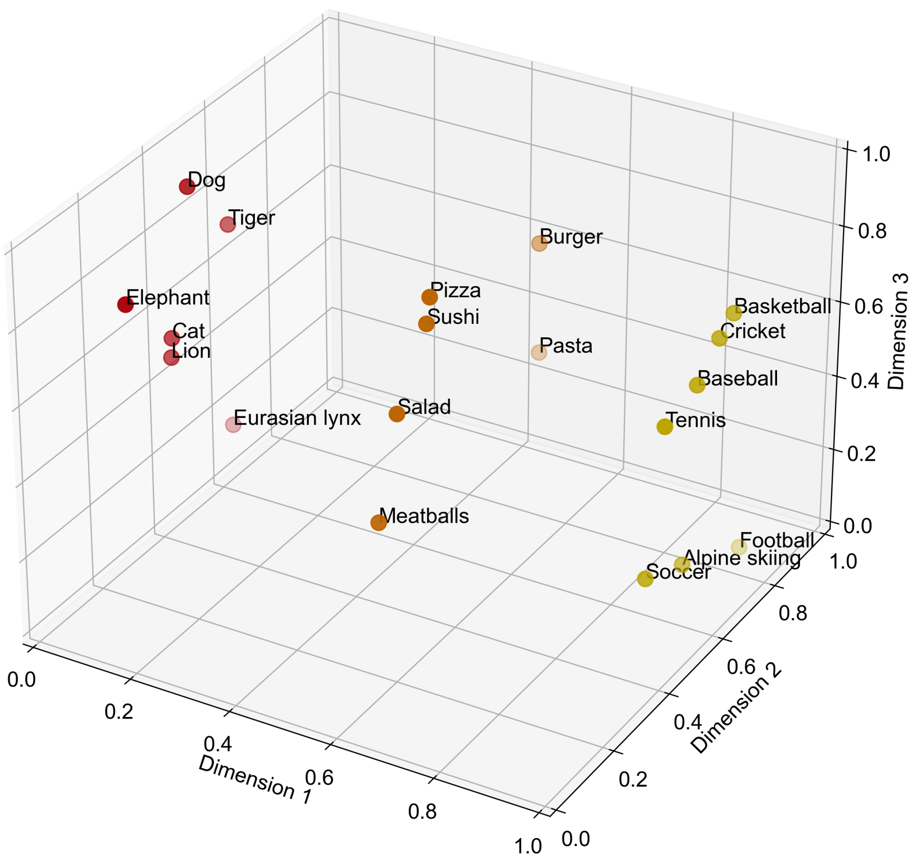

Figure 2.4: Simplified 3D space with simplified embeddings for various words

There are numerous types of models and techniques that have been developed to efficiently compute vector embeddings. One such is Continuous Bag-of-Words (CBOW) and Skip-gram models, introduced by Mikolov et al. [41]. These were early yet foundational methods for generating word embeddings. These models leverage large corpora to predict tokens from their context CBOW, or with the context from the tokens (Skip-gram). Both of these techniques leverage the trained models ability to learn semantic and syntactic nuances of the tokens in the training corpus [41, 42].

Advancements in vector embedding technologies enhance NLP tasks such as text classification and sentiment analysis. Embedding models that can process language or images with nuance and precision are crucial for accurate real-world applications [43].

There are new Embedding functions released often, built-upon different language models and employing various different techniques. The evaluation of these embedding functions often remains constrained to a narrow set of tasks. Muennighoff et al. [44] tried to address this issue by introducing Massive Text Embedding Benchmark (MTEB), which spans 8 embedding tasks covering a total of 58 datasets and 112 languages. The leaderboard is currently actively maintained on hunggingface *.

Two models that rank highly on the leaderboard is Salesforce's open source *SFR-Embedding-Mistral* model which exemplifies advancements in embedding technology for text retrieval tasks [45]. Similarly, OpenAI has developed several closed source embedding models that also rank highly on the MTEB leaderboard [46, 47].

Embeddings are often used to compare documents against each other, or against a given user query. This is often done by computing similarity scores between words, phrases, or documents, which are represented as vectors in the embedding space. These scores quantify the closeness, or "similarity" between different texts.

The similarity between two vector representations is typically measured using the cosine similarity metric. This calculates the cosine of the angle between two vectors. This metric ranges from -1 (the exact opposite document) to 1 (the exact same document), with 0 indicating orthogonality (no similarity). The cosine similarity $\text{sim}(u, v)$ between two vectors u and v is defined as:

$$\text{sim}(u, v) = \frac{u \cdot v}{\|u\| \|v\|} \tag{2.6}$$

where $u \cdot v$ is the dot product of the vectors u and v, and $\|u\|$ and $\|v\|$ are the Euclidean norms of both vectors.

*Massive Text Embedding Benchmark (MTEB) Leaderboard on Huggingface (accessed on May 1, 2024)

2.8 RAG

RAG is the process of integrating retrieval mechanisms into the generative models. This approach effectively combines the strengths of both retrieval and generative language modelling to enhance a model's ability to accurately recall factual information by utilising an external knowledge base during the generation process [2].

RAG was developed to address the limitations of large pre-trained language models that could compress a large training corpus into its weights, but could struggle with accessing and precisely manipulating this information when required. The term "hallucination" would come to describe the event where models would "recall" incorrect information, as shown in figure 2.5.

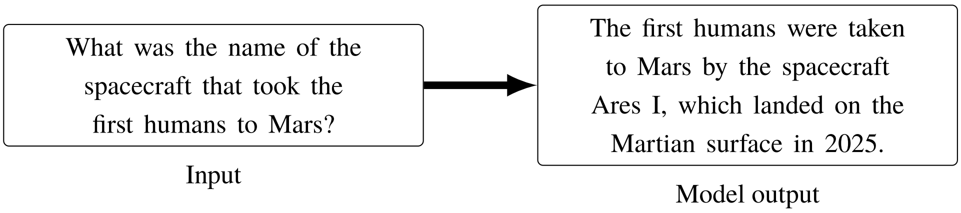

Figure 2.5: An example of how a model can hallucinate an answer to a question.

The fact that language models tend to have a propensity to hallucinate, and the simple fact that it is very time consuming to train language models, mean these models would often lag behind in knowledge-intensive applications where correctness is crucial. The integration of a non-parametric memory, or an external knowledge base, allows these models to retrieve relevant information during the generation process and thereby producing more accurate responses [2].

In a typical RAG setup, the systems architecture is split into two main components: the retriever and the generator. The retriever is a language model trained to search and fetch relevant documents. Nowadays, this process often utilises dense vector representations of documents (see section 2.7.2) which enables efficient and effective search [48].

The returned documents are then fed into the generator, this is a seq2seq model, which then synthesises the information into coherent text. The generator is often instructed to only use authoritative facts that are returned from the retriever and not rely on its internal knowledge for factual statements. This dual-component approach allows RAG to dynamically access a large corpus of knowledge while maintaining its ability to generate fluent and contextually appropriate language [2]. This method has shown significant improvement over a purely parametric-approach in various tasks such as question answering and fact verification [23, 49].

When google announced Gemini 1.5 [3] they claimed it could effectively recall knowledge over prompts as large as a million tokens. It remains to be seen if the hallucination, training-time and context length problems can be overcomed and remove the need for a RAG system when building knowledge-intensive applications on-top of a LLM.

2.9 AI Assistants

AI assistants is a type of AI-system that is designed to support human users by performing tasks that typically require human intelligence. Stuart Russell and Peter Norvig wrote in *Artificial Intelligence: A Modern Approach* [50] that an assistant should interact with their environment to achieve specific goals rationally and effectively.

AI assistants must be good at NLP for effective communication with humans in addition to good knowledge representation such as with a RAG toolchain. The assistant must also possess good reasoning and decision-making abilities, as those exhibited by a modern LLM. An assistant should also utilise machine learning to improve from user interactions.

2.10 Measuring usability and acceptance of new technologies

To assess how effectively a user can interact with a technology, for instance, an AI assistant, Jakob Nielsen's "Usability Engineering" [51] is a seminal book that defines usability in terms of learnability, efficiency, memorability, safety, and satisfaction. All of these can be measured through specific metrics. The IBM Computer Usability Satisfaction Questionnaires, developed by Lewis [52], offer a tool that's been validated through the years to measure these dimensions.

Interactions with AI agents through conversation is very affected by the agent's ability to engage in social dialogue. Bickmore and Cassell [53] discuss the importance of dialogue in building engagement long-term between users and conversational agents. Their conversational agents communicated over the phone, but their framework for understanding the qualitative feedback from users about their experiences can also be applied with an AI assistant.

Technology Acceptance Model (TAM) was introduced by Davis [54] and is particularly relevant for examining the acceptance of new technologies such as AI assistants. TAM suggests that perceived usefulness and ease of use are key factors for whether a new technology is accepted and used. The model is useful for investigating users' attitudes towards the utility and usability of new technologies, not the least of which is AI and AI assistants.

Expectation-Confirmation Model (ECM) was introduced by Bhattacherjee [55] in 2001 and it extends the understanding of user satisfaction beyond initial acceptance which is outlined in TAM. ECM includes user expectations, perceived performance, and confirmation of expectations into the satisfaction assessment. This model is especially useful in assessing whether a technology meets or exceeds the users' expectations over time.

2.11 Related Work

The introduction of RAG by Lewis et al. has been crucial for the development of NLP systems. The model they proposed has been shown to combine the generative power of a LLM with the factual accuracy of verified knowledge bases. From a research perspective, RAG is a method of addressing "hallucinations" in generative models [2]. "Hallucinations" refers to instances where a LLM asserts a fact that is provably untrue as though it were accurate, see figure 2.5. With the techniques proposed by Lewis et al. a system can be constructed around the model that ensures only verified facts are used to answer input prompts.

The research in this paper focuses on applying different LLMs and RAG techniques to a real domain. The paper by Lewis et al. is also more abstract and proposes general terminology and concepts. This thesis implements those concepts in real software deployed in a real domain. Additionally, this thesis experiments with various practical RAG techniques. Furthermore, it proposes a method for measuring the effectiveness of these techniques by introducing feedback questions into the system. This approach is also deployed in the experiments conducted within this thesis.

SciBERT was introduced by Beltagy et al.[56]. *SciBERT* is a version of *BERT* [19] which is specifically trained on scientific texts. This corpus of scientific texts contains scientific publications which is meant to enhance the models performance on NLP tasks such as sentence classification, and entity recognition within the scientific domain.

There are numerous similar models developed for other domains. Even though it may appear that *SciBERT* was developed for similar reasons to the system in this thesis. *SciBERT* was adapting a language model, albeit a smaller one, to a new domain. However, adapting a LLM using RAG is a different approach since there is no training on any new data. The model in that case is simply adapted by constructing prompts, and attaching an external knowledge-base, see section 2.8 for more information.

José Hernández-Orallo wrote[57] about different ways of assessing various methods that evaluate AI models. The methods he evaluates range from traditional task-oriented approaches to ability-oriented approaches. Task-oriented puts emphasis on solving specific problems. Within NLP and LLMs, this can be stuff like Stanford Question Answering Dataset [58] or parts of MMLU [8].

Ability-oriented tests on the other hand, measure cognitive abilities of AI systems and models using similar methodologies to those done on humans. Human intelligence has for a long time been measured using various different intelligence and algorithmic tests. Examples of such tests are traditional IQ-tests or calculating Kolmogorov complexity.

The research in this thesis is narrower than what is proposed by Hernández-Orallo. His paper takes a broader approach on AI and AI evaluation. Even though the motivation for this thesis is similar, evaluation methods and benchmarks for LLMs, RAG and AI generally are too task-oriented. As described earlier in this chapter, considering the size of the training data for LLMs, one argument is that the models *learn* solve problems simply by memorising the solution. The solution explored by Hernández-Orallo, and this thesis, is a more abstract evaluation method. In the case of this thesis, it's by measuring user preferences, as opposed to constructing a new test-method, as proposed by Hernández-Orallo.

Zhang et al. [59], introduced a framework, called *CoGenesis*, for making small and large language models collaborate. This framework works by deploying smaller models on devices and larger models on compute clouds. The goal with developing a framework like this is to produce a system that can balance privacy and performance.

Their evaluation involves using a synthetic dataset with an enriched personal context on-top of two open-source datasets. Their framework is benchmarked against standalone LLMs using various metrics. In contrast to the research in this thesis, the aim with their research is purely to find a way to address the privacy shortcomings of current approaches for building LLM applications. They have similar metrics like relevance in their benchmarking, but also measure privacy protection by measuring how much user-specific context information is kept on local models while only sharing generalised instructions with the cloud based LLMs.

Siriwardhana et al. [11] write about RAG-end2end. This is an enhancement to RAG models that enables joint training of the retriever and generator. This improves adapting the base model for open domain question answering. Their paper evaluates their new approach against *COVID-19* datasets. Their suggested model shows significant improvements against traditional RAG.

The paper by Siriwardhana et al. is very similar to the research in this thesis. They both try to evaluate RAG techniques. The key difference is that Siriwardhana et al. try to construct a new paradigm, whereas this thesis only evaluates existing paradigms against each other. This thesis presents no new RAG paradigms.

This thesis also creates a new dataset for the evaluation by integrating the question into custom built software that is deployed into a real domain. In their paper, Siriwardhana et al. use existing QA-pairs to evaluate their model. Their paper uses significantly larger datasets than constructed in this thesis. Albeit the domain for the research in this thesis is much more novel, and much less common in the training data for the LLMs evaluated.

To summarise, while there is existing research that explores and evaluates various enhancements and frameworks for connecting LLMs with different RAG approaches, this thesis evaluates some of these techniques in a practical, real world domain. This thesis also introduces a method for measuring the effectiveness of different RAG techniques that are common when building AI assistants. The method relies upon integrated feedback questions that measure more nuanced properties of AI assistant's usefulness for users in the selected domain.

Chapter 3

Method or Methods

This chapter outlines the methodologies and procedures used for conducting the research described in this thesis. The focus is on presenting the chosen methods for data collection, analysis, and evaluation of the deployment and effectiveness of AI-assistants within Canvas course rooms at KTH.

3.1 Research Process

The research process within this thesis consisted of three main phases. This section will outline each phase and what the purpose was.

3.1.1 Prof of concept

The study will aim to achieve an assistant using open source and permissively licensed LLMs and RAG techniques, that is comparable to the current best in class models provided under less permissive licesnses and are only available through properitery APIs. Considering that this is a fairly complex task, the research started with a long phase of constructing various proof of concepts.

There would be two main proof of concepts, one using proprietary models and RAG techniques, and one strictly using software that is under open source licenses and can be self-hosted.

3.1.2 Implementation of study software

The second phase consists of constructing the software that will be used during the study. This software is the actual AI assistant and all its components, such as the course room crawler and indexer, the infrastructure to run LLMs at scale and a Graphical user interface (GUI) to interact with the assistant. This software also needs to be able to exchange various different components that are the subject for the study, such as the LLM being used in a chat. Furthermore, the software needs to be able to record user interactions and responses to questions.

3.1.3 Conduct the study at KTH

This phase consists of deploying and monitoring the assistant in the course rooms that have enrolled in the study. This might involve modifying the software such that it works in the new course room, or ensuring that there is enough capacity on the platform to sustain the new users.

3.1.4 Analyse results

Once the study has concluded the data analysis will be conducted, the final results and analysis can be found in chapter 4. The various different data points collected in the experiments were all stored in a database. Once the study concluded, the analysis was done using different python jupyter notebooks. The notebooks can be found in the github repository for this thesis *. There are three notebooks, one for each theme.

The first is the *"usage"* notebook. This contains the charts and tables constructed to analyse the general usage of the assistant in all courses. These cover metrics like, how many messages were sent, how many chats were held, users registered in each course, etc. The second notebook is the *"feedback"* notebook. This notebook contains an analysis of all user submitted responses to the feedback questions. The third and last notebook is the *"performance"* notebook. This notebook contains charts and tables that analyse all performance metrics collected by the system, such as how quickly models produced response to prompts, how long indexing took, etc.

3.2 Research Paradigm

This research in this thesis follows a pragmatic approach that blends aspects of the study from positivist paradigms to investigate the practical application and user reception of AI assistants generally within the educational setting. Additionally, it seeks to determine which technologies perform

*github.com/nattvara/DA231X/results/notebooks best by analysing segmented responses to user queries based on the specific technologies used to generate the answers. The pragmatic approach supports using mixed methods to answer the research questions effectively, focusing on ’what works’ as the basis for knowledge claims. The positivist elements of the study will quantify which technologies deliver the fastest responses and yield the most usage among users. Participants will be exposed to one technology from a predefined set, selected through random sampling. This methodological approach allows for a positivist analysis to determine which technology is the fastest and most preferred by its users.

The collection and analysis of the feedback from the participants in the study is an interpretivisit approach to answer the questions of more subjective nature. These are questions like which technologies or models generate the most accurate results, or answers that are more preferred by users.

To summarise, initially the chatbot is deployed (positivist approach), followed by the collection and analysis of user feedback (interpretivist approach) to understand the broader implications of AI-assistant technology in specialised domains such as education. This method aims to understanding the functional capabilities of the AI-assistant in addition to its practical utility and acceptance by end-users, students and teachers.

3.3 Data Collection

The data collection in this study comes from two sources;

- The AI-assistant software built to conduct this research. This includes metrics from the usage of the system, in addition to integrated survey components of the system such as responses to questions from users of the system.
- Responses to coursework questions from students in selected courses that are part of the study, submitted as part of their course requirements.

3.3.1 Data collected by the software constructed for this study

The AI-assistant software was constructed to record various pieces of data, these can be grouped into four categories, operational data, usage data, performance data, feedback data. The data collected by the AI-assistant is anonymous. No personal information was recorded by the system, aside from any information that may have been submitted as part of a question by the user to the assistant.

3.3.1.1 Operational data

The operational data refers to data collected by the system to function. This includes information in or about the course rooms such as the various pieces of content found in a course room and their relations. This can be lecture slides, lab assignments, links to external sites etc.

3.3.1.2 Usage data

Usage data in system refers to data that is generated when users use the system. This includes session information and its metadata, chat information such as which course room a chat is associated with and which messages were sent by the user and the AI-assistant in each chat.

3.3.1.3 Performance data

Various metrics regarding the performance of the system are also computed and kept. These include metrics such as how quickly a model generates a response to a given prompt and how long the response. Also, how quickly a vector embedding was computed and how quickly the index returns documents.

3.3.1.4 Feedback data

The feedback data is the most intentional data tracked by the system. This includes questions injected into the chat at specified intervals. These questions and intervals are the same for all users. The questions have a predefined set of answers and the system tracks which of the answers a user selects, or if they don't select any answer at all. In addition binary thumbs up/down questions are also asked about certain responses from the system.

3.3.2 Data collected by questions in the coursework

In the courses that agreed, a form was distributed to all their students. The form was designed to gather insights as to why students replied like they did in the feedback questions. The form was slightly different in each course depending on the demographic of the participating students. For instance, one course was designed for maths teachers, therefore their form included questions meant to gather insights from the teaching viewpoint. The forms, their questions and responses, can be viewed in their entirety in Appendix B, Appendix C and Appendix D. The qualitative analysis was very high-level, as it was not the primary form of gathering data for this study. The qualitative analysis can be found in section 4.3.

3.3.3 The participants in the study

The study sources its participants from courses from course administrators that have volunteered for their courses to participate in the study. The courses were found by emailing course responsibles at the EECS school at KTH in addition to connections of the supervisors of this thesis. Table 3.1 shows the courses the participating students were taking.

Course Code	Course Name	Credits (hp)	Number of Students
LD1000	Lär dig lära online	2.0	83
DD1380	Javaprogrammering för Pythonprogrammerare	1.5	**393**
MG2040	Assembly Technology	6.0	32
LD1006	Kognitiv psykologi för lärare: Matematikundervisning	3.0	51
DD1349	Projektuppgift i introduktion till datalogi	3.0	213
DD2419	Project Course in Robotics and Autonomous Systems	**9.0**	42
DD1367	Software Engineering in Project Form	**9.0**	231
	Total		**1045**

Table 3.1: Courses that participated in the study, number of credits per course, and the number of students enrolled

3.4 Experimental design and Planned Measurements

Participants in the study will be randomly assigned to groups, each of which will use a specific set of technologies and techniques. This random assignment will be managed by software written for this study, and it will apply to every chat session started with the assistant by the participating student. Each group will utilise a unique configuration based on one of these predefined parameters;

- The language model used to run the chat. This is the LLM that is used to generate chat replies within the chat. It's also used for the internal logic of the assistant. This internal logic consists of tasks such as identifying whether the user asked the assistant a question that needs data from the knowledgebase.

- The RAG technique and technology used to access the indexed data. This could be TF-IDF or an embedding function. The latter group also has a defined embedding model assigned.
- Post processing of the retrieved documents. This is a boolean flag that configures the assistant to post process documents before inserting them into the context window for the configured LLM to generate an answer with.

The software allows for questions to be inserted into a chat the participant is having using the following two triggers.

1. After a participant has had n chats and is sending the m:th message in that chat
2. After a student clicks one of the frequently asked questions

Each of these triggers allows for inserting any given number of questions into the chat. Any question has to follow one of the following templates;

a) A question, such as *"Was this a good answer?"* accompanied with a "thumbs up" or a "thumbs down" button to answer the question with.

b) A question, such as *"How accurate did you find this answer?"* with a set of answers to select from such as *Very accurate*, *Somewhat accurate*, *Neither accurate nor inaccurate*, *Somewhat inaccurate*, *Very inaccurate*.

With these configurations, the experiment aims to measure the impact of the predefined parameters on users' responses to the questions posed during the chats at the configured triggers.

3.5 Test environment

3.5.1 Software

To reproduce the results of this study the software that was written to crawl course rooms, index their content and host the chat with the assistant is available in its entirety on github, see Appendix A.

3.5.2 Configuration

There is quite a bit of configuration needed to get the software operational. The *README* in the source code extensively covers how to run the software in most common environments, see Appendix A.

3.5.3 Data and access to canvas

To run the AI assistant with data from an actual course room the bot needs to have access to a canvas through a KTH registered user. Due to time constraints the option to use the official Canvas API was abandoned early in the planning of this study. The softwares' crawler therefore needs the cookies of an authenticated user with access to the course rooms included in the study.

3.5.4 Models

The software constructed for the study utilise the transformers python library [*] maintained by HuggingFace. The library manages the download and loading of the open source LLMs and embedding models used in the thesis. This also means that if the models are no longer available on the huggingface registry, or the registry is nonoperational, the models have to be obtained by other means. The open source models that are supported by the software are the following models

- The Mistral-7B-Instruct, provided by MistralAI, [†]
- The Gemma-7B, provided by Google, [‡]
- The Falcon-7B, provided by TII UAE, [§]
- The SFR-Embedding-Mistral, provided by Salesforce, [¶]
- The Meta-Llama-3-8B-Instruct, provided by Meta, [‖]

[*] The GitHub page for the transformers library /github.com/huggingface/transformers (accessed on July 4, 2024)

[†] huggingface.co/mistralai/Mistral-7B-Instruct-v0.2 (accessed on July 4, 2024)

[‡] huggingface.co/google/gemma-7b (accessed on July 4, 2024)

[§] huggingface.co/tiiuae/falcon-7b (accessed on July 4, 2024)

[¶] huggingface.co/Salesforce/SFR-Embedding-Mistral (accessed on July 4, 2024)

[‖] huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct (accessed on July 4, 2024)

In addition to the open source models the software also supports experiments using some proprietary models by OpenAI. The two models that are supported are the *GPT-4* model and the *text-embedding-3-large* embedding model [*]. Both of these are accessible using the python API [†] which require an OpenAI subscription and API key.

3.5.5 Hardware

The study software should be able to execute on most common hardware and most parts of the application are not particularly compute intensive. The notable exception to this is the worker processes in the LLM Service part of the software that, depending on the model, run quite intensive compute loads. That is unless the agent runs either of the proprietary cloud hosted models provided by OpenAI.

If the agent is running one of the LLMs it can run, such as the *Mistral 7B instruct* model, the agent needs access to a quite capable GPU. For the supported models this needs to be a GPU with at least 24 GB of memory. Example of such graphics cards are *NVIDIA GeForce RTX 3090*[‡], *NVIDIA TITAN RTX*[§] or *NVIDIA A10 Tensor Core*[¶]. This used servers on AWS, specifically the G5 instances (g5.4xlarge) equipped with *NVIDIA A10 Tensor Core* GPUs [‖]. The open source embedding models such as *SFR-Embedding-Mistral* can be run on CPUs, which only require the same amount of RAM available to run the models.

3.6 Assessing reliability and validity of the data collected

3.6.1 Validity of method

To research how well AI-assistants work in a specialised domain, other methods could've been used. For instance, instead of building an assistant and deploying it, the thesis could've explored similar domains and drawn

[*] platform.openai.com/docs/guides/embeddings/ (accessed on July 4, 2024)

[†] github.com/openai/openai-python (accessed on July 4, 2024)

[‡] nvidia.com/sv-se/geforce/graphics-cards/30-series/rtx-3090-3090ti/ (accessed on July 4, 2024)

[§] nvidia.com/en-eu/deep-learning-ai/products/titan-rtx/ (accessed on July 4, 2024)

[¶] nvidia.com/en-us/data-center/products/a10-gpu/ (accessed on July 4, 2024)

[‖] aws.amazon.com/ec2/instance-types/g5/ (accessed on July 4, 2024) conclusions from the success of similar systems in similar domains. However, actually implementing an assistant and evaluating its efficacy using standard methods such as TAM and ECM is a more accurate way of measuring the research questions laid out.

3.6.2 Reliability of method

The methods used in this study, including those for measuring user acceptance of the tool, are considered reliable. However, the results of the study will be heavily impacted by the type of users using the tool. Given the research is taking place at the division of robotics at a technical university, the participating courses are mostly technical courses with tech savvy students. To reproduce the results of this study it would have to closely reproduce the student population participating in the study. To increase the reliability of the method in this study the research could have been performed at a wider array of universities with a more diverse student population.

3.7 System documentation

Appendix A includes links to the source code developed for the research in this thesis. The source code includes a *README* with comprehensive instructions for how to build, run and deploy the software.

3.8 Proof of Concepts

The following section will outline the various Proof-of-concept (POC) applications that were built before the actual software that was written to conduct the research outlined in this thesis. Even though it's difficult to distinctly separate POCs from each other, in total around 9 POCs were produced. Each POC will outline what it was trying to accomplish and what the outcome was. Some POCs have been omitted since they were either dead-ends, or simply too similar to one of the other POCs described below.

3.8.1 Langchain based applications

Langchain is a company [*] and framework [†] for building context aware reasoning applications. The framework allows for easy composition of language models and RAG techniques and tools that makes it easy to build chatbots with a connected knowledge base. This section outlines some POCs that were made with the langchain framework.

3.8.1.1 GPT-4 and text-embedding-3-large

To build a chat application with an AI-assistant that has access to an external knowledge base, one of the most popular approaches is to use langchain to connect the following four parts.

1. A LLM, such as GPT-4 to run the chat.
2. A LLM, such as GPT-4 to run the query construction.
3. An embedding function, such as OpenAI's text-embedding-3-large used to index and query documents.
4. A vector store, such as ChromaDB, that stores the vector embeddings and associated documents [‡]

.

In this configuration, Lanchain acts as the glue connecting these components and handling tasks like chunking larger documents. The goal of this POC was to test a common approach for building AI assistants and

[*] langchain.com (accessed on July 4, 2024)
[†] python.langchain.com (accessed on July 4, 2024)
[‡] trychroma.com (accessed on July 4, 2024)

evaluate its potential for use in the full study. A video can be seen here that showcases this POC.

3.8.1.2 Mistral 7B v0.2 and e5-large-v2

There was a POC constructed that had the same approach as the one outlined in 3.8.1.1 with the notable requirement that all tools had to be under an open source licence. This meant the GPT-4 model and text-embedding-3-large models couldn't be used. A similar version of the same POC was made that used the Mistral 7B v0.2 model and the embedding function e5-large-v2 [60]. These are both under an open source licence and are freely available on Huggingface * †. This POC did however suffer from poor performance in initial tests for retrieval and performance. It was difficult to tune the prompts to get decent performance. This POC showed it was difficult for the researcher to get good performance out of certain models using the langchain framework.

3.8.2 Custom applications

This section outlines some major and minor POCs that were made without any frameworks that are popular LLM applications, aside from very common python libraries such as PyTorch and Hugginface's transformers library.

3.8.2.1 Simple Python API for models on Huggingface

Langchain and similar tools support running language models locally. However, working with the prompt templates in less advanced models than GPT-4 and achieving good retrieval and chat performance was challenging. Therefore, a simple POC was developed to create higher-level Python abstraction APIs on top of the Hugging Face Transformers library that could be integrated into completely custom solutions. These APIs include examples like those shown in listings 3.1 and 3.2. A short video can be seen here that demonstrates a chat application (without an integrated knowledge base) built on-top of these simple APIs.

*huggingface.co/mistralai/Mistral-7B-Instruct-v0.2 (accessed on July 4, 2024)
†huggingface.co/intfloat/e5-large-v2 (accessed on July 4, 2024)

Listing 3.1 High level API on-top of Huggingface's tranformers library that can be used for generating text using models available on Huggingface.

```python
def load_hf_model(
    model_path: str,
    device: str
) -> (transformers.AutoModelForCausalLM, transformers.AutoTokenizer):
    """
    Loads a Hugging Face causal language model and its tokenizer for a given
    model path and device.
    """

def generate_text(
    model: transformers.AutoModelForCausalLM,
    tokenizer: transformers.AutoTokenizer,
    device: str,
    params: Params,
    prompt: str
) -> str:
    """
    Generates text from a prompt using the specified model, tokenizer, and
    generation parameters.
    """

async def generate_text_streaming(
    model: transformers.AutoModelForCausalLM,
    tokenizer: transformers.AutoTokenizer,
    device: str,
    params: Params,
    prompt: str
) -> AsyncGenerator[str, None]:
    """
    Asynchronously generates text from a prompt, yielding tokens incrementally.
    Useful for streaming responses.
    """

def _tokenise_inputs(
    tokeniser: transformers.AutoTokenizer,
    input_texts: list[str],
    max_length: int = 8192
) -> dict:
    """
    Tokenizes the input texts with padding and truncation.
    """

def should_stop_generating(
    output_token_ids: list,
    tokenizer: transformers.AutoTokenizer,
    params: Params,
    token_id: int
) -> bool:
    """
    Determines whether to stop generating text based on stop conditions.
    """
```

Listing 3.2 High level API on-top of Huggingface's tranformers library that can be used for generating vector embeddings using models available on Huggingface.

```
def load_hf_embedding_model(
    model_path: str,
    device: str
) -> (torch.nn.Module, transformers.AutoTokenizer):
    """
    Loads a Hugging Face embedding model and its tokenizer for a given model
    path and device.
    """

async def compute_embedding(
    model: torch.nn.Module,
    tokeniser: transformers.AutoTokenizer,
    text: str
) -> List[float]:
    """
    Computes and returns the normalized embedding for a given text using the
    specified model and tokenizer.
    """

def _compute_model_embeddings(
    model: torch.nn.Module,
    tokenised_inputs: dict
) -> torch.Tensor:
    """
    Computes the model embeddings from the tokenized inputs.
    """
```

3.8.2.2 Mistral 7B v0.2 and Opensearch

The goal with this POC was to build a version of the RAG application that didn't use a vector embedding function. Instead, this POC would utilise a traditional search service such as Elasticsearch or Opensearch. These implement "traditional" search algorithms such as TF-IDF, as outlined in section 2.7.1. This POC was very easy to implement and showed great promise.

3.8.2.3 Post processing for smaller models

When working with models that don't produce the best scores on public benchmarks, such as MMLU, there are a number of techniques that can be employed that could improve the performance of a RAG system. These generally smaller models suffer from worse scores on precision and recall benchmarks. This means they are worse at recalling facts injected into the conversation by a RAG pipeline. One of the techniques that can be used is post-processing retrieved documents before they are inserted into the chat. There are a number of ways of achieving this. One of the techniques that was tried, and eventually implemented in the final study, was to use a post-processing mechanism, where each of the retrieved documents are passed through a post-processing function, which reduce the size of the document, as shown in Equation 3.1 and Equation 3.2.

$$R = \text{LLM}\left(Q, \{D_i\}_{i=1}^{N}\right) \tag{3.1}$$

Where:

- R is the generated response.
- LLM is the language model function.
- Q is the user query.
- $\{D_i\}_{i=1}^{N}$ are the matching documents retrieved from the index.

$$R = \text{LLM}\left(Q, \{\text{PP}(D_i, P)\}_{i=1}^{N}\right) \tag{3.2}$$

Where:

- PP is the post-processing function.
- P is the post-processing prompt.

There are different strategies for the prompt that reduce the size of the document. This prompt can instruct the language model to extract quotes from the document related to the query, summarise key facts related to the query, or a number of other methods. The one that was chosen for the final study was extracting quotes as this showed the most potential. Figure 3.1 and Figure 3.2 illustrate this process.

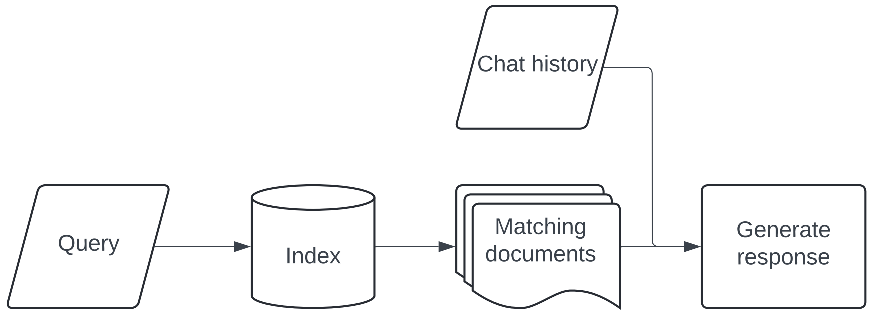

Figure 3.1: Diagram that shows how documents are injected into the chat, without post-processing

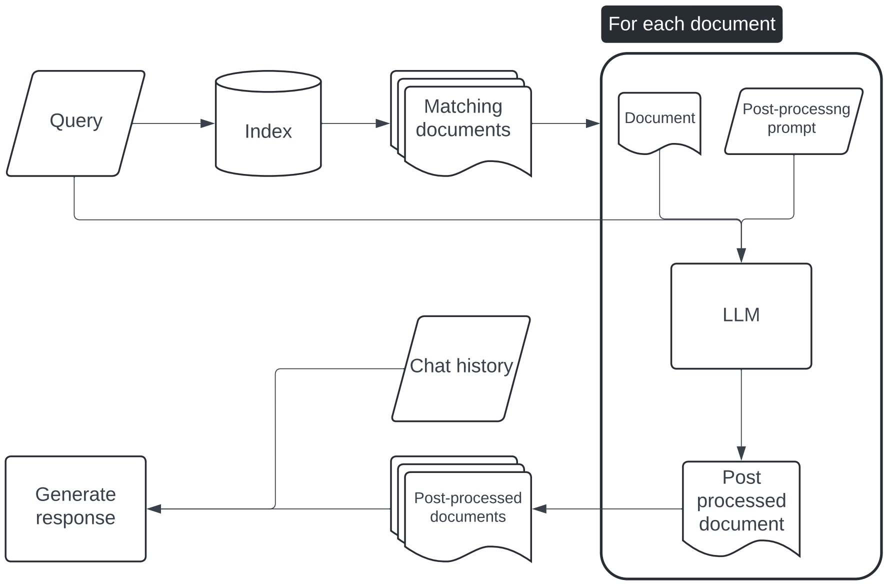

Figure 3.2: Diagram that shows how documents are post-processed before they are injected into the chat

3.8.2.4 Too much processing

Processing the documents to be smaller in size such that smaller models could accurately recall facts from retrieved documents was investigated until the point where additional processing would no longer improve the quality of the answers,

There were additional efforts put towards investigating the processing of documents to be smaller in size. The goal was to enable smaller models to accurately recall facts from the retrieved documents. This process continued until additional processing no longer improved the quality of the answers. There were POCs produced that for instance processed the documents during the indexing phase in addition to the retrieval phase. During the indexing the documents were compiled into smaller "facts" that were indexed on their own. However, reducing the documents in size decreased the retrieval algorithms, both TF-IDF algorithms and embedding functions.

Some processing however could improve the performance. Such as including a summary of the entire document with each chunk of that document. In summary, pre-processing the documents, and chunks of each document, during the indexing process, did increase the system's ability to produce accurate answers to user queries. However, processing the documents too much was found to eventually lead to a reduction in accuracy.

3.9 The architecture of the software

This section will outline the final software that was constructed to investigate the research question in this thesis. The section will describe the goal of the software, what components it consists of, how each component works and showcase how it helps students get answers to their questions.

3.9.1 What the purpose of the software is

The purpose of the software is to crawl Canvas course rooms and index their data into a knowledge base. The software should expose this knowledge base in a chat based application integrated into the same course room it has crawled. The tool should be able to randomly sample a configuration of models and tools for indexing, retrieval and chat between users of the tool. Finally, the tool should inject questions into the chat and track the necessary data points to investigate the research question of this thesis outlined in section 1.2.1.

3.9.2 What the software can test

The software was constructed to measure how certain variables would impact the user feedback to certain configured questions. Section 4.2.2 shows all questions that were used, and what possible answers existed. The variables that were supported can be seen in Table 3.2, Table 3.3 and Table 3.4. The students did not know which configuration they used, nor that there existed different configuraitons. The table also includes a column if this variable was actually used in the experiment, or if it was just implemented but never enabled in the experiments.

Variable	**Supported**	**Enabled**
GPT-4 by *OpenAI*	Yes	Yes
Mistral 7B Instruct v0.2 by *MistralAI*	Yes	Yes
Llama 3 8B Instruct by *Meta*	Yes	No

Table 3.2: The models supported by the software and if they were used in the study

Variable	**Supported**	**Enabled**
Full text search	Yes	No
Vector search using *SFR Embedding Mistral* by *Salesforce*	Yes	No
Vector search using *text-embedding-3-large* by *OpenAI*	Yes	Yes

Table 3.3: The indices supported by the software and if they were used in the study

Variable	**Supported**	**Enabled**
Full text search	Yes	No
Vector search using *SFR Embedding Mistral* by *Salesforce*	Yes	No
Vector search using *text-embedding-3-large* by *OpenAI*	Yes	No

Table 3.4: The indices supported by the software **with** post-processing enabled, and if they were used in the study

3.9.3 Overall architecture

The software is primarily written in Python, with a service-based architecture. This means it is divided into distinct domain specific services, each handling specific domains and functionalities. These services handle one of four things, LLM-, Download-, Index- or Chat-functionality.

These services are written in such a way that they can be incorporated into any executable within the project. There are numerous executables within the application, these are;

- A graphical user interface, which is a web-based application.
- A Representational State Transfer (REST)ful HTTP API.
- A websocket server.
- A job-runner that execute background tasks from a queue.
- A worker node for the LLM service that keep a LLM in-memory ready to generate a response to a prompt for the given model.

Figure 3.3 shows an overview of the architecture of the software written to conduct the research and which components are called by other components.

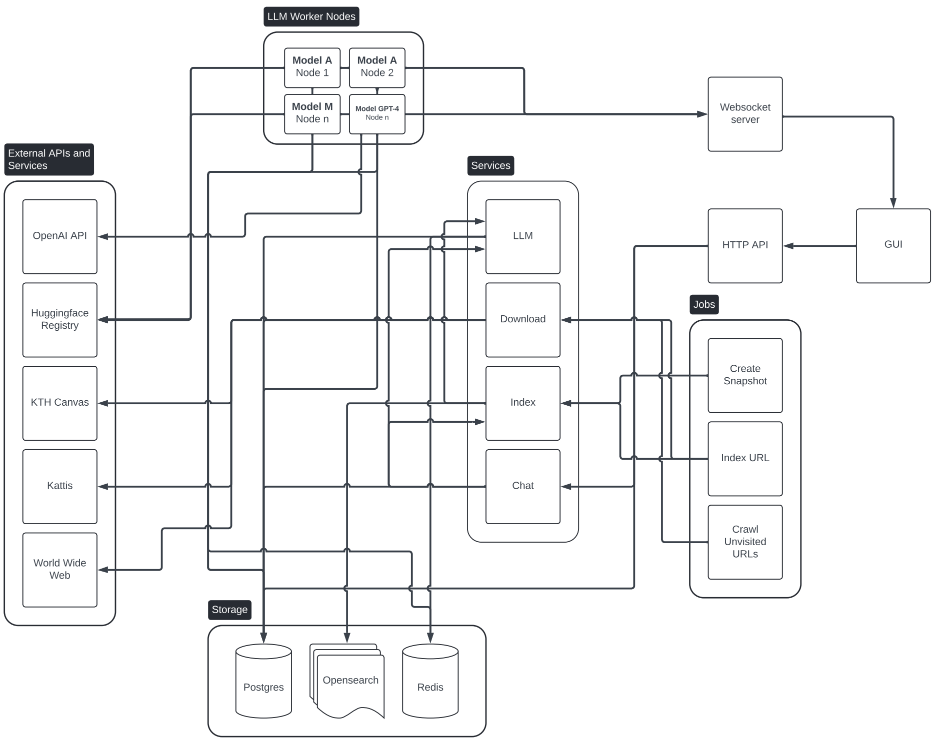

Figure 3.3: Diagram that shows the system architecture of the software constructed to run the study

3.9.4 Dependencies

The software is written on-top of numerous Python and Javascript libraries, these are documented in their entirety in the systems source code repository. In addition, the software needs the following major dependencies to operate.

- A Postgres database.
- An Opensearch index.
- A Redis in-memory storage.

3.9.5 Courseroom Crawler

The software contains a job that is run every minute, which checks if a new snapshot should be taken of a course room. A snapshot contains all Uniform Resource Locator (URL)s and their content that was crawled from a course room. A piece of content may be a Canvas webpage, a file hosted on Canvas, an external url or file etc.

If a new snapshot is created of a course room the crawler will immediately crawl the course room. The crawler has global rules it follows for all course rooms, certain pages are ignored and how some content should be indexed. For instance, some common tools, such as the programming assignment tester Kattis * have a known format. The crawler employs specific crawlers for these common sites and tools to ensure a high-level of data quality. The crawler was built upon the Playwright framework developed by Microsoft †. This is a browser automation framework, mostly used for end-to-end testing of web applications. Using a browser to crawl websites is more compute-intensive than simply requesting the html files without rendering them. The benefit of using a browser, and actually rendering the content of the pages, is that content that's requested by scripts on the page gets loaded and can be extracted.

Ideally, the public canvas HTTP API ‡ should be used to ensure the highest level of data quality and a more reliable connection. However, this API had been disabled by KTH IT and would've taken time and resources to get access to.

---

*More information about how kattis is used at universities can be found here kattis.com/universities (accessed on July 4, 2024)

†playwright.dev (accessed on July 4, 2024)

‡Documentation for the API can be found here canvas.instructure.com/doc/api/ (accessed on July 4, 2024)

3.9.6 Document index

Opensearch, which started as a fork of the popular search engine Elasticsearch, serves as a full text search backend and document store for the software. The opensearch server also stores each vector embedding for all supported vector embedding functions used in the study. Opensearch is also used in the software to compute similarity scores for the computed vector embeddings of all documents, utilising the k-NN algorithm for efficient nearest neighbour search. This enables rapid retrieval and ranking of relevant documents based on their vector similarities which is one of the key research goals of this thesis. Listing 3.3 shows an example of a document indexed in a snapshot of a course room that participated in the study.

Listing 3.3 Example of an indexed document from the course *DD1367 Software Engineering in Project Form 9.0 credits* which participated in the study.

```
{
  "name": "Communication guidelines: DD1367 HT23 Programvarukonstruktion...",
  "text": "Document chunk: 1/1\nDocument summary: The document provides...",
  "text_raw": "Due to the mass amount of daily emails, I would like to...",
  "url": "https://canvas.kth.se/courses/43604/modules/items/765971",
  "sfr_embedding_mistral": [
    0.012644807808101177,
    -0.007494120392948389,
    0.005445912480354309,
    ...
  ],
  "text_embedding_3_large": [
    -0.009533963166177273,
    0.01081753708422184,
    -0.01858356036245823,
    ...
  ]
}
```

3.9.7 Running large language models at scale

One of the principal requirements for the study software was the ability to investigate the impact of different LLM on the experience of the user. This could impact the recall abilities of the system due to the models involvement in both understanding what the user is searching for and for summarising that information. Additionally, large embedding models would also be investigated within the scope of the study.

Proprietary models, such as those provided by OpenAI and are studied in this thesis, are always executed on the model vendors infrastructure and are accessible only via an HTTP API. These APIs are not compute-intensive for the consuming application, such as the one developed for this thesis. However, open-source models, like the Mistral family of models, are available for anyone to run. One of the goals of this thesis was to investigate the feasibility of running such applications with all necessary dependencies on-premise. This involved executing these models within the application. This presents an engineering challenge, as these models are not very mature yet, and the infrastructure for running them is not well developed.

For the software in this study an *LLM-service* was developed. This service presented a high-level API that could be used by any part of the application, such as the chat-service, for producing messages, or the indexing service, for producing summaries or embeddings. The API had a high-level function that took a model name, model parameters and prompt to execute, and returned a prompt handle as a response. A prompt handle, in addition to the provided arguments, kept the necessary queuing information for the system to function, some performance metrics and the eventual output of the model.

In the background the service provides the infrastructure for organising a virtually unbounded number of worker-nodes for each model. Any worker node keeps the model running in memory, which could be either in CPU or GPU memory. Loading the model into memory is a time-consuming operation, which was one of the driving reasons for this approach. This architecture allows for scaling the service to keep up-with-higher demand for prompt-completions. Heavy load could be caused by for instance, many users using the system simultaneously, or indexing one or more course rooms simultaneously.

Organising systems like this have many difficulties, particularly regarding the distributed nature of this architecture and the compute intensiveness of model inference. For instance, since the execution time of a prompt can be upwards of a minute, the worker nodes need the ability to stream the response back to an end user in real-time token-by-token, to provide a good user experience. For this reason a websocket service was implemented, that allows the end-user application to listen for tokens being produced over a websocket that the producing node is directly connected to.

3.9.8 Assigning a chat configuration to a user

The system supports defining multiple chat configurations, each consisting of a specific set of tools and models for the user's chat experience. A chat configuration consists of;

- **A Language Model**: This is the LLM used for generating assistant responses and computing internal agent functionality such as the post processing of documents, if that's enabled for the config.
- **Retrieval Technique**: The method used for information retrieval, either an embedding model or full-text search.
- **Embedding Model**: Specifies the embedding model if used.
- **Document Processing**: Indicates whether documents should be post-processed before being inserted into the chat context (see section 3.8.2.3).

Any configuration can be selectively enabled or disabled based on the experimental requirements.

When a user accesses the system for the first time, a configuration is randomly assigned to ensure an unbiased distribution among users. Once assigned, the configuration remains consistent for the lifetime of that user's account. This is however only true if the user uses the same browser, a new browser would result in a separate account.

3.9.9 Gathering user feedback

The software was designed to facilitate user feedback collection. This feedback is gathered through a fairly flexible feedback entity, which integrates questions in the configured language of the course. A feedback questions was triggered by specific conditions, such as:

- After the response for the n-th message is sent in the m-th chat a user has had, where n represent the number of messages in the current chat and m the number of chat a user have had in total.
- After generating a response to a frequently asked question.

These triggers were chosen to capture the user experience over a certain period, enabling control over the duration of system usage when analysing the users response to each feedback question.

The feedback question format was designed to be extensive. There were two formats implemented for the research in this thesis. These formats were multiple choice questions, and binary *thumbs up / thumbs down* questions. Figure 3.4 and 3.5 showcase these two formats respectively.

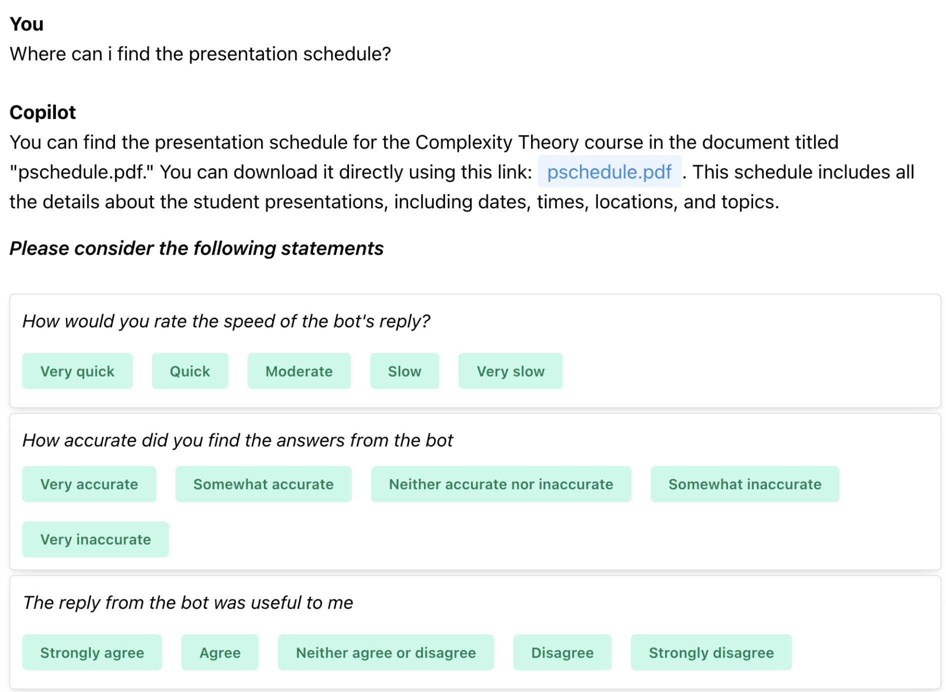

Figure 3.4: Example of a number of multiple choice questions inserted into the chat

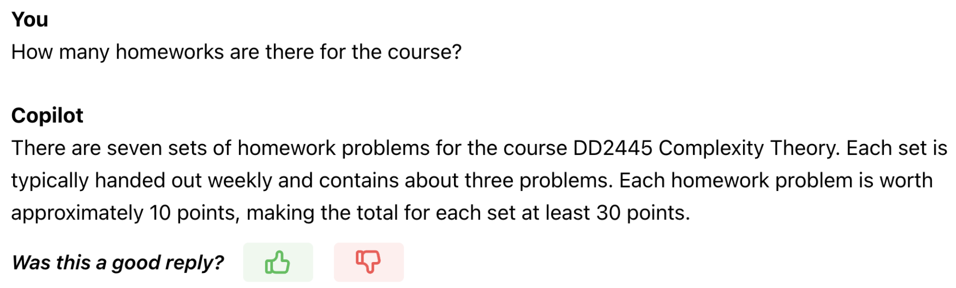

Figure 3.5: Example of a thumbs up/thumbs down question inserted into the chat after the user clicks on a FAQ.

3.9.10 User interface

The assistant built for the system was designed to live within the LMS used at KTH, known as Canvas. Each course in canvas is assigned a course room. This course room contains all the information a participant in the course might need, such as course announcement, assignments and modules. Figure 3.6 shows the course room of one of the courses that participated in the study. The GUI for the assistant designed for this thesis, was designed to live within the courseroom as an iframe.

The software constructed for this study was designed to integrate with KTH's LMS, Canvas. Each course in Canvas has a designated course room that contains all information for students in the course. This includes announcements, assignments, modules and more. Figure 3.6 shows the course room of one of the participating courses in the study. The GUI for the assistant, developed for this thesis, was embedded within the course room as an iframe, accessible through the course sidebar in the web, and navigation menu in the Canvas mobile app.

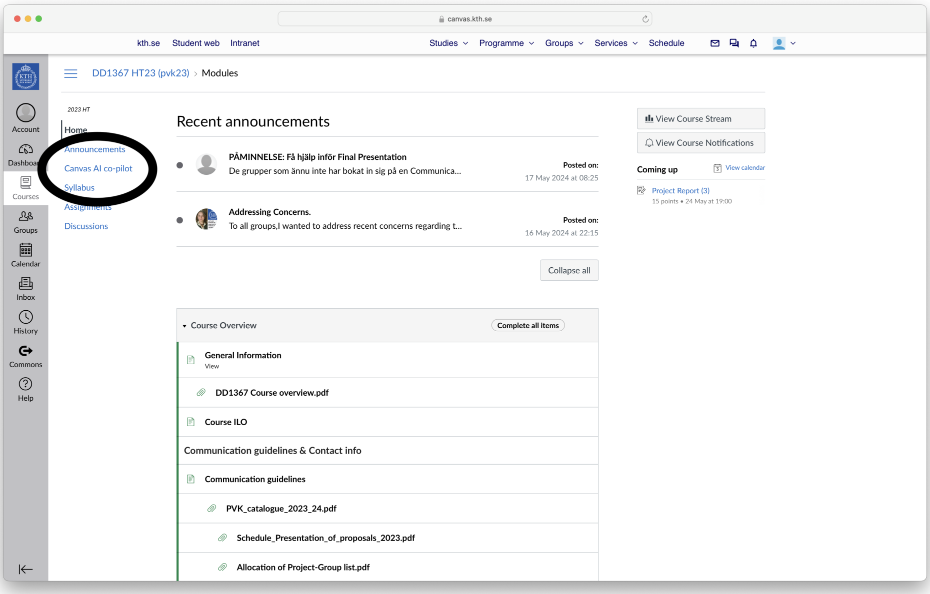

Figure 3.6: The course room of DD1367 Software Engineering in Project Form 9.0 credits in canvas

The system utilise a script that compiles the most common questions. These are displayed each time a user starts the application, as can be seen in figure 3.7. This allows the user to quickly understand what other students have asked the tool, and quickly start a conversation.

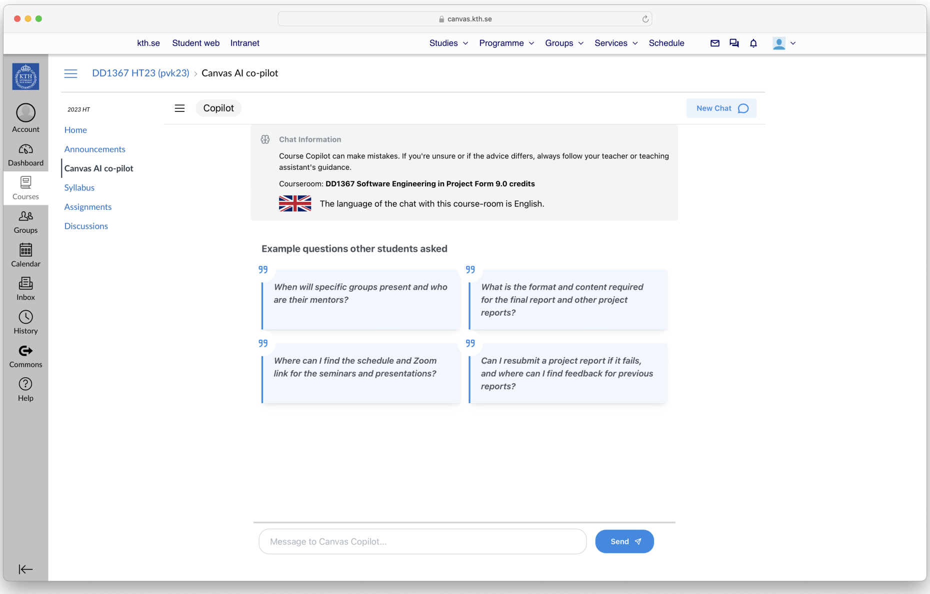

Figure 3.7: The chat UI with frequently asked questions shown before the user sends their first message

Overall, the GUI of the tool is very simple as can be seen in figure 3.8. It allows users to send and receive messages from the tool. The responses from the assistant support some rich formatting, such as bold fonts. Links to cited documents are highlighted and forwards the user to the corresponding page or file in canvas when clicked. A video demonstration of the software being used by a student can be seen here.

Any teacher or TA can be granted administrative privileges which yields them access to a list of all chats that have been had between students and the assistant in their course. This allows them insights into what the chatbot is responding to the students' questions.

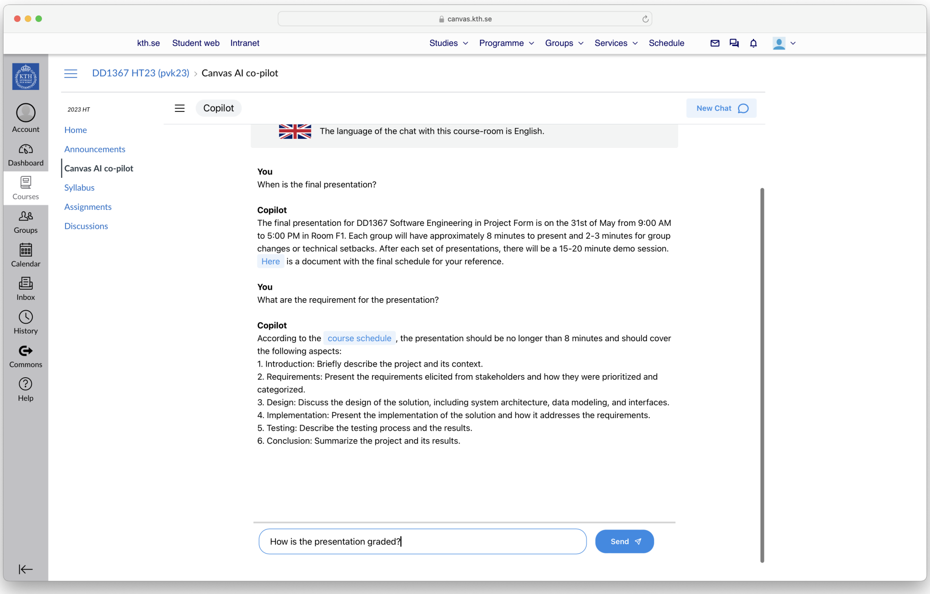

Figure 3.8: A short conversation with the assistant

3.10 How the software is deployed

Considering the nature of running LLMs at scale the software had quite compute-intensive requirements. Due to sponsorship from KTH Innovation there were credits accessible on Amazon AWS that were available to use for the study. This meant AWS was a suitable cloud infrastructure provider to deploy the software that was built for the study.

AWS has hundreds of services of varying levels of abstraction. The requirements on the infrastructure for this study was primarily that it had to support GPU heavy compute-workloads. Since the software was built to be packed into Docker images, the infrastructure had to support connecting GPU devices to the running Docker containers. AWS Amazon Elastic Container Service (ECS) supports creating *"Task Definitions"* and executing them in services that are hosted on their fully managed *Fargate* platform, or hosting them on regular Amazon Elastic Compute Cloud (EC2)-instances. The latter has a wide array of instance definitions to choose from, of which multiple has one or more GPU devices connected. The ECS platform was therefore chosen as the main deployment platform for the software. The entire system architecture can be seen in figure 3.9.

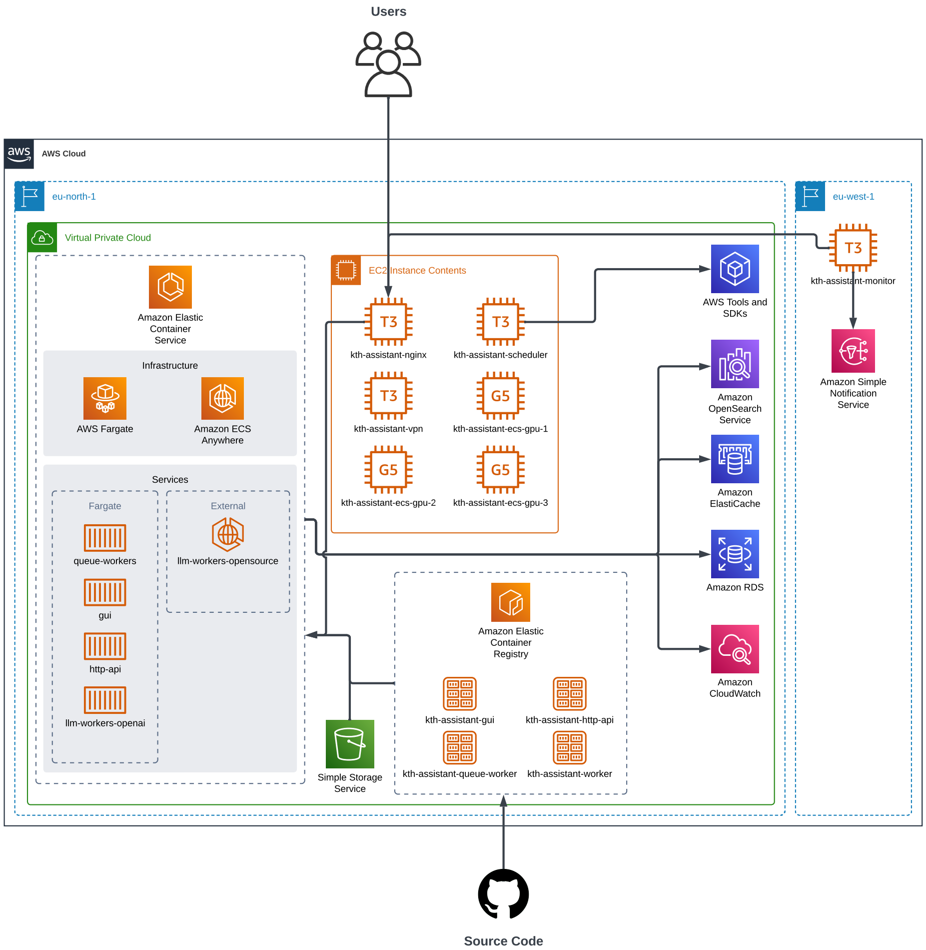

Figure 3.9: Diagram that shows how the software was deployed on Amazon AWS

In the repository for the source code there are a few github workflows, one of which builds the executables for the project. The executables are four docker images, these are:

- **HTTP API:** This docker image runs the HTTP and Websocket servers for the software.
- **GUI:** This runs a small node server that servers the user interface for the software.
- **LLM Worker:** This runs a worker node for a given LLM in the system

- **Queue Worker:** This executes a pool of worker nodes for the jobs that can be dispatched in the system

Each of these images are published to their respective Amazon Elastic Container Registry (ECR)-registry. The ECR are themselves used as source for the task definitions in ECS. Together with configuration stored in Amazon Simple Storage Service (S3) the image and task definition are deployed into services in ECS. These services can be scaled independently of each other, and use different hosting environments depending on their workload.

Each service can be scaled manually or put in auto scaling groups. The only service that would see any considerable load is the *LLM-worker* containers. However, since they need to keep the LLM in-memory in order to provide real time answers to users, it can't be scaled using traditional metrics such as CPU or memory usage. No sophisticated auto-scaling was configured, since it was deemed unnecessary for this study. Instead, the study relied upon a schedule maintained by a simple script that during certain peak hours scaled up the infrastructure necessary to run more nodes for each enabled LLM or embedding function.

Traditional services in AWS were used for running the respective component of the application. One postgres server was deployed in Amazon Relational Database Service (RDS), an in-memory Redis cache was created in *Amazon ElastiCache* and an OpenSearch cluster was created in *Amazon OpenSearch Service*. Logs from all services and software components were collected in *Amazon CloudWatch*.

Whilst the service was operational keeping track of whether the system was operational became a priority. During the initial development and deployment it became necessary to track if the service was up and running and be notified of any outages. Therefore a simple playwright script was built that simulated a student entering a question in each course room on a rotating schedule. This would publish a message on a Amazon Simple Notification Service (SNS) topic that would trigger an email if a response couldn't be produced by the system.

Chapter 4

Results and Analysis

This chapter will present and analyse the results of the research conducted in this thesis.

4.1 Feasibility of building an AI assistant on open source technologies

One of the goals of the research in this thesis was, as outlined in section 1.4, to assess the feasibility of building an AI-assistant on open-source technologies and deploying the agent in an academic setting. This section will outline the results and showcase the impact open source tooling had on the implementation of the AI assistant.

4.1.1 How popular was the system

The system was developed during the spring of 2024 and gradually deployed to seven real courses at KTH starting on the 18th of April 2024. The students in the courses that participated in the study held a total of 656 chats and the users of the system sent 2373 messages. As can be seen in Figure 4.1 and Figure 4.2 these steadily increased over the course of the study as students initiated new chats with the assistant.

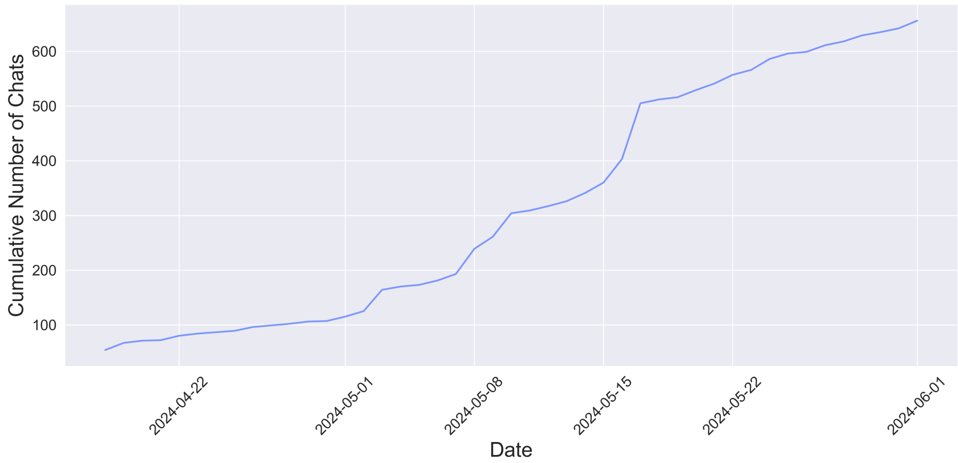

Figure 4.1: Cumulative number of chats started by users participating in the study

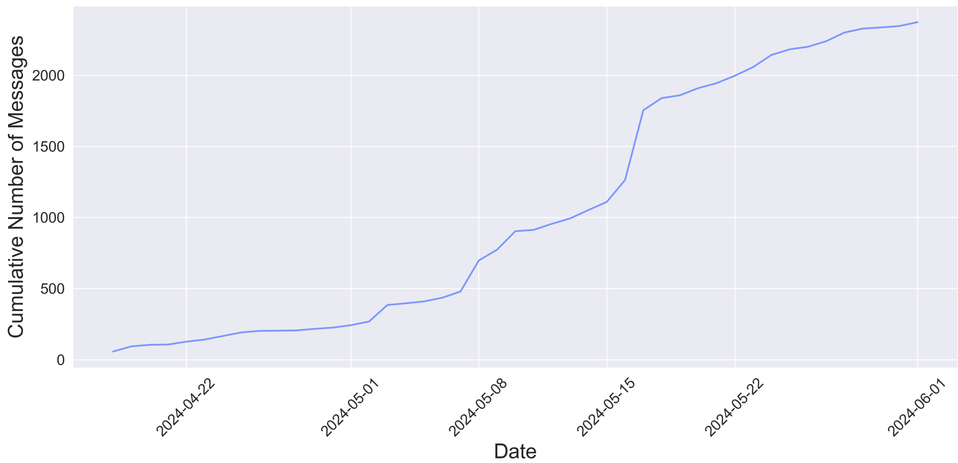

Figure 4.2: Cumulative number of messages per day

Separating the chats initiated in the separate course rooms we observe that some courses followed a fairly linear increase in the number of chats. One example of this is the course *MG2040 Assembly Technology 6.0 credits*, which can be seen in figure 4.3.

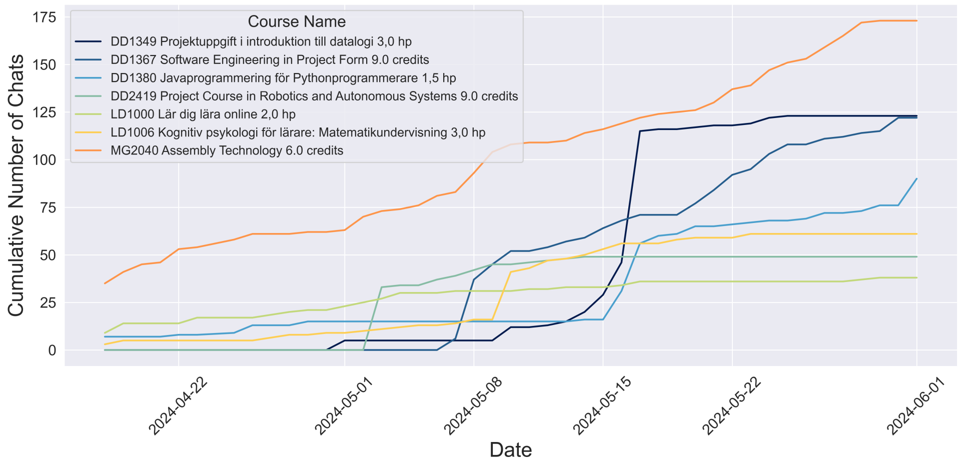

Figure 4.3: Cumulative number of chats started by users participating in the study in each course

Looking at other courses in figure 4.3 it is evident that not all courses follow the same pattern as *MG2040*. Some courses initially have very few chats due to the chatbot not being deployed simultaneously across all courses. Table 4.1 details the start dates for each course. For instance, *DD1349 Projektuppgift i introduktion till datalogi 3,0 hp* exhibits a steep increase in users when it launched, followed by no further growth. This is because the course officially ended shortly after the chatbot was introduced. In all courses participating in the study, the chatbot was deployed well after the courses had already begun, therefore a similar pattern can be seen in many other courses.

Course	**Go live date**
LD1000 Lär dig lära online 2,0 hp	2024-04-17
LD1006 Kognitiv psykologi för lärare: Matematikundervisning 3,0 hp	2024-04-17
MG2040 Assembly Technology 6.0 credits	2024-04-17
DD1349 Projektuppgift i introduktion till datalogi 3,0 hp	2024-05-01
DD2419 Project Course in Robotics and Autonomous Systems 9.0 credits	2024-05-03
DD1367 Software Engineering in Project Form 9.0 credits	2024-05-07
DD1380 Javaprogrammering för Pythonprogrammerare 1,5 hp	2024-05-13

Table 4.1: The start dates for each course, when the bot was deployed in each canvas course room.

Figure 4.4 shows the total number of chats held in each course, and Figure 4.5 shows how these were distributed over each course over time. The course *MG2040* held the most, 173 chats. *DD1349* held the second most and *DD1367* the third most, 123 and 122 chats respectively.

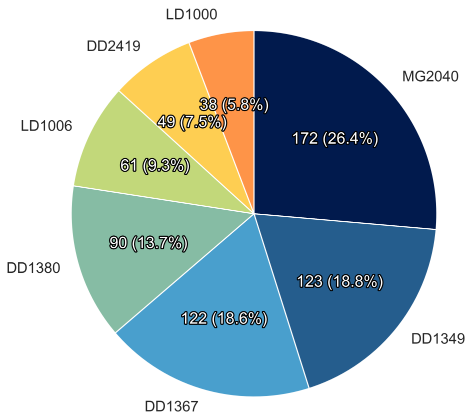

Figure 4.4: Number of chats held by in each course

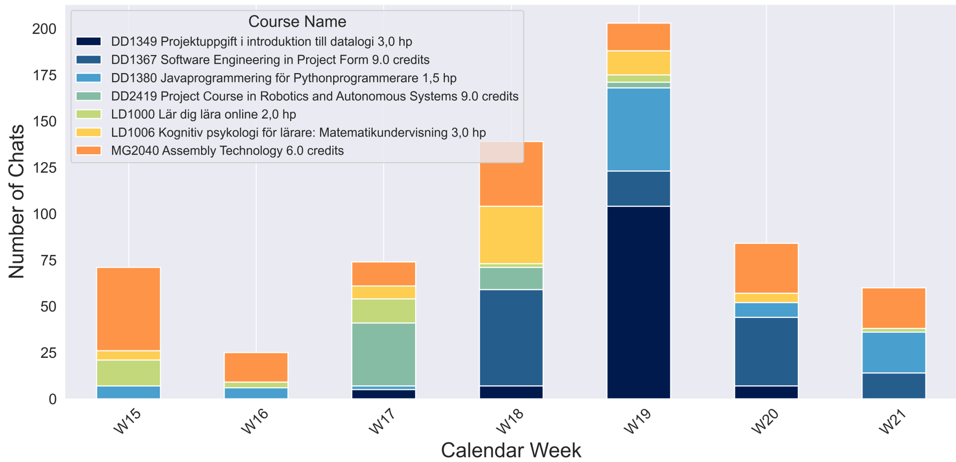

Figure 4.5: Number of chats held each week per course

Looking at the number of sessions created in Figure 4.6, we can see a similar linear pattern. A session is started whenever a user loads the application without already having loaded it before. A session is not tracked between devices, therefore a user would have two sessions if the same user accessed the chat on two different devices, such as a desktop and a mobile phone. However, the same session is used across courses. Looking at the breakdown per course in Figure 4.7, we can see that this is likely the case for the course *MG2040 Assembly Technology*.

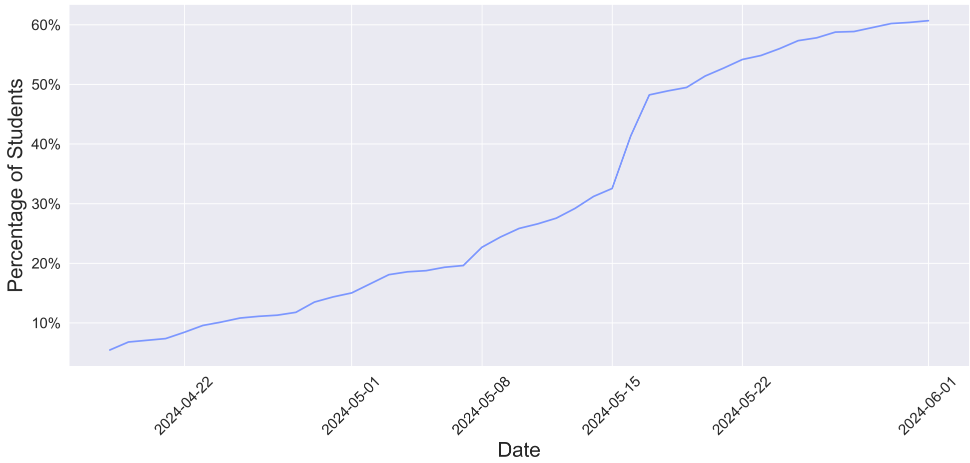

Figure 4.6: Cumulative number of users as percentage of number of users in all participating courses that had started a chat

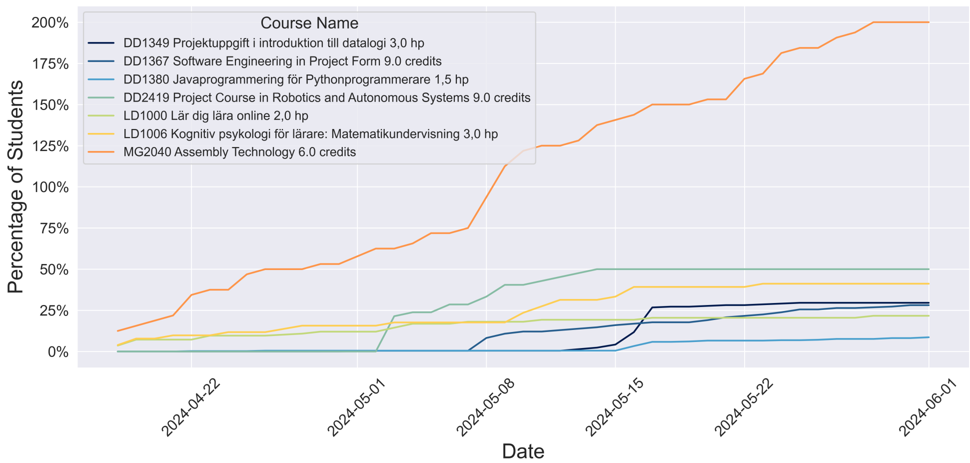

Figure 4.7: Cumulative number of users per day in each course, as a percentage of the course’s students, that had started a chat

Looking at *MG2040*, it looks like nearly 200% of its 32 students have started a session. This discrepancy is likely due to the following two reasons,

1. The inclusion of teachers and TAs who also use the service. In a small course, this significantly skews the percentages that can be seen in Figure 4.7.

2. Sessions are counted uniquely across different devices. This factor likely affects other courses as well, although it is most evident in *MG2040*. This was unfortunate, as it impacted how effectively the experiment could track the number of users in each course. However, this was the best and easiest way to keep the participants anonymous in the study.

Looking at the distribution of how many chats and messages is sent per session, as seen in figure Figure 4.8 and Figure 4.9 we can see that it was very common for users to only start one or two chats. Most users sent quite a few messages though. The average user held 2.28 chats and sent 9.65 messages.

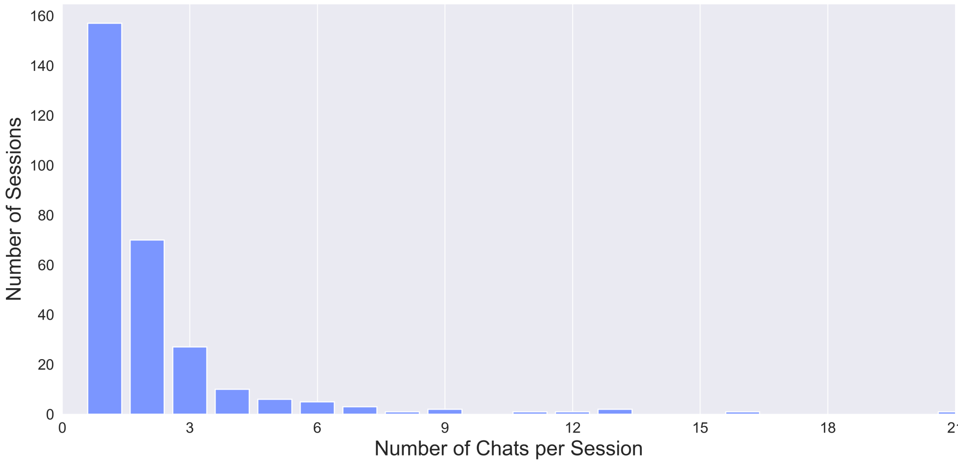

Figure 4.8: Number of sessions that held a given number of chats

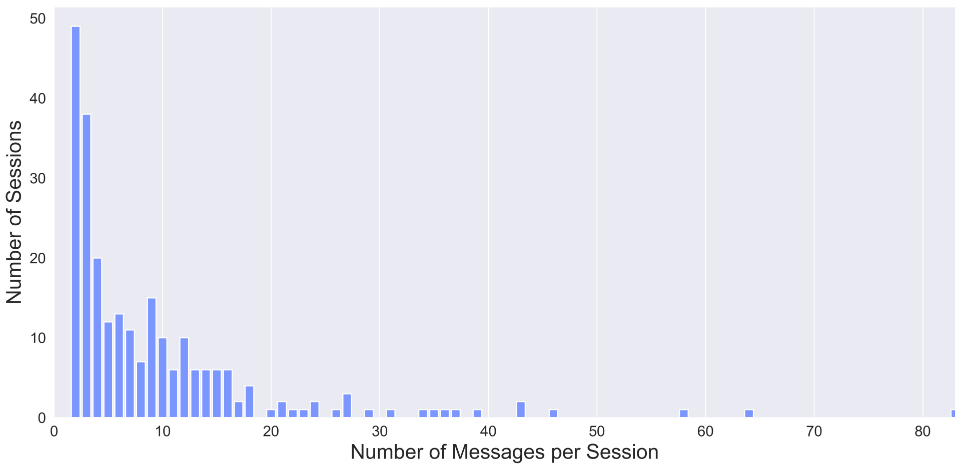

Figure 4.9: Number of sessions where a user sent a given number of messages

4.1.2 Open source v. Proprietary LLMs

With regards to the feasibility of building AI assistants on open source technologies there are a number of metrics to look at for comparing open source LLMs to proprietary language models. Table 4.2 shows metrics for both models that were included in the experiment. The experiment was designed to sample between the included models randomly, and as we can see the number of sessions started between the two models are virtually the same. However, there are notable differences in the number of chats started and messages sent between the two. The proprietary model, *GPT-4* by OpenAI, leads the open source model, *Mistral-7B-Instruct-v0.2* by MistralAI. Section 4.3 and 4.2 will showcase the user preferences with respect to these models, which could explain the discrepancy between two models with regards to simple usage metrics, which is what is shown in Table 4.2.

Model	No. Sessions	%	No. Chats	%	No. Messages	%
openai/gpt4	**319**	**50.39**	**386**	**58.84**	**1238**	**52.17**
mistralai/Mistral-7B-Instruct-v0.2	314	49.61	270	41.16	1135	47.83

Table 4.2: Statistics of Sessions, Chats, and Messages by Model

Looking at the operational performance of both models included in Table 4.2, there are two notable metrics that were measured in the experiment with respect to operating these models, more specifically metrics that doesn't measure the quality of their responses (these are covered in). The metrics are; the response time for the model and time taken to generate queries. The latter is measuring what is generated by the system to query the index that was produced when crawling the course room. The LLM is obviously used to generate the assistant's next message, but it is also used to generate a search query, based upon the current conversation between the assistant and the user. The time taken to generate this query is obviously important for the overall performance of the system.

Figure 4.10 shows the daily average response for each model. This includes the time taken before a worker node had picked up the workload. This is important because in the event of high traffic to the system, LLM tasks could be queued up and response times could increase. The chart shows that the two models are generally very similar in terms of the time it takes them to produce a reply to the user's question. It is notable however, that the open source alternative (*Mistral-7B-Instruct-v0.2*) has higher peaks on certain days.

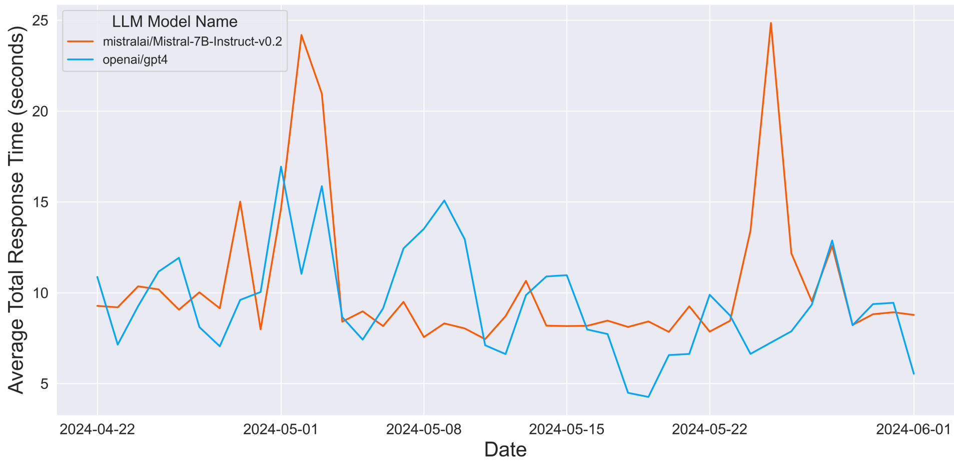

Figure 4.10: How long each model took to generate a response, including time spent pending.

Figure 4.11 shows the time taken to produce queries. Similar to what could be said about Figure 4.10, both models perform very similarly. However, in this metric the open source alternative is faster. The reason the open source model outperforms *GPT-4* on this metric is likely due to the higher latency sometimes observed on the OpenAI API. The custom built solution to operate LLM for this study exhibits much lower latency.

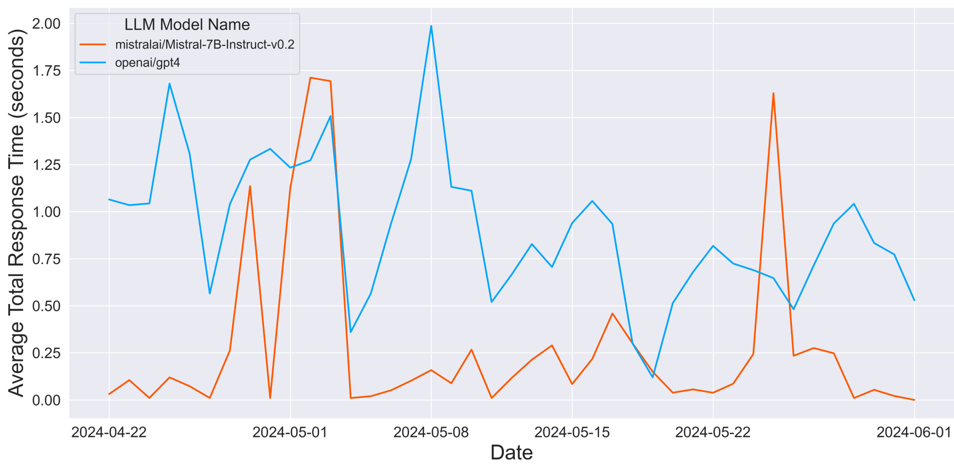

Figure 4.11: How long each model took to generate queries.

4.1.3 Open source v. Proprietary Embedding functions

The intention of the research in this thesis was to compare open source embedding functions with proprietary alternatives commonly used in RAG-based applications. In addition to this, the experiment was also designed to be able to measure vector search as a retrieval technique, with traditional full text search. However, due to the number of students that participated in the study, no configuration was used that didn't use the vector search strategy with the embedding function *text-embedding-3-large* by OpenAI [47]. So no metrics were captured in the retrieval phase for any other strategy or model. The retrieval time taken for the model that was used is shown in Figure 4.12. Metrics were, however, captured during the indexing phase on other models.

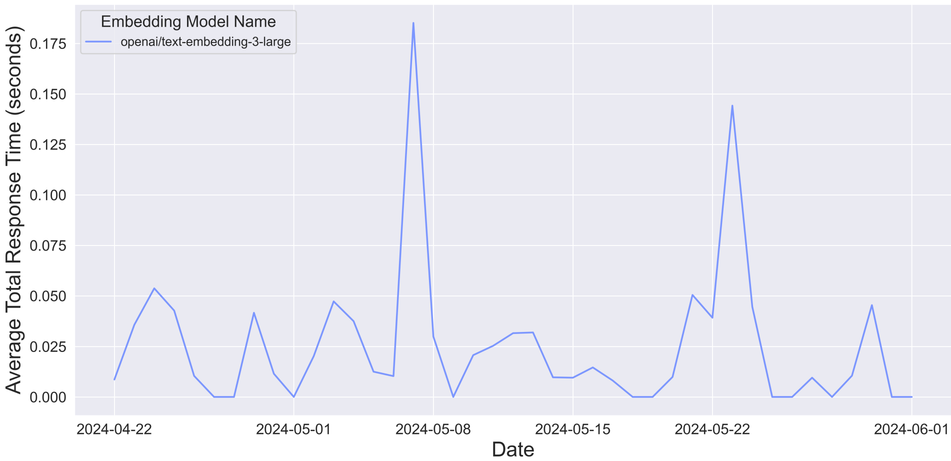

Figure 4.12: How long each model took to execute queries.

4.1.3.1 Understanding the indexing

The indexing of course rooms was done regularly. It wasn't done on a schedule, it was instead done on an ad-hoc basis whenever a course was updated with new content. Figure 4.13 shows a timeline for each course which includes the date for when each snapshot of the course room was taken.

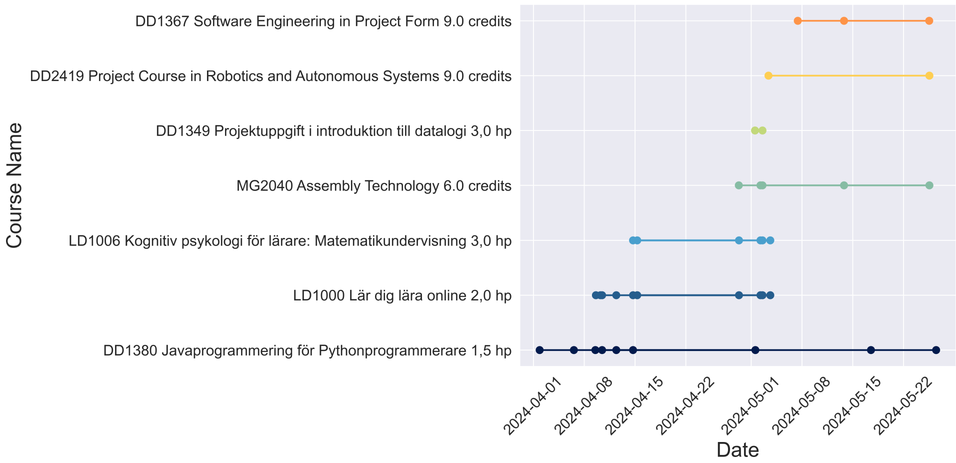

Figure 4.13: Timeline for each snapshot taken of the courses participating in the study.

Each course room is different and includes pages and content of varying length. No exact metric for how much content was included in each course room is presented in this section, however Figure 4.14 shows how many urls the crawler found in each course, and how many of them were indexed. This is an imperfect, yet decent proxy for how "large" a course room was.

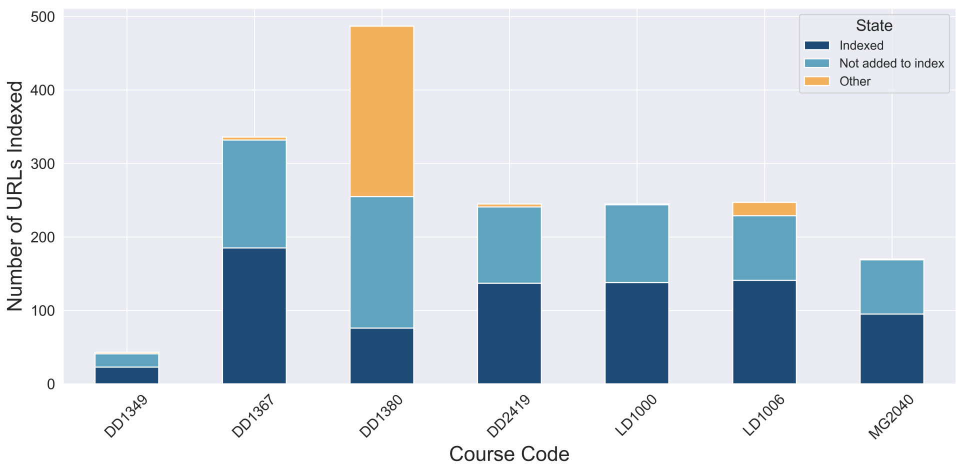

Figure 4.14: Number of URLs included in the most recent snapshot of each course.

4.1.3.2 Measuring indexing performance

The time taken by indexing course rooms varied widely depending on the load on the system. If more than one course room were indexed simultaneously, indexing took much longer. The most time consuming part of indexing a course room was computing the embeddings for all documents added to the index. Table 4.3 shows the open source embedding function used in the experiment, *Salesforce's SFR-Embedding-Mistral*[45], which was chosen because it had the highest score on the MTEB-benchmark*, is two orders of magnitude slower than *text-embedding-3-large*, the currently best embedding function developed by OpenAI. The reason for this was likely the execution environment chosen for the open source candidate.

Model Name	Average Response Time (seconds)
Salesforce/SFR-Embedding-Mistral	101.59s
openai/text-embedding-3-large	**0.03**s

Table 4.3: Average Total Response Time per Embedding Function

During the experiment the embedding functions utilised the same servers that ran the open source LLMs. To utilise the hardware rented for this thesis optimally, the open source embedding models were executed on the unutilised CPUs of the servers which ran the LLMs on their attached GPU devices. The embedding models are, as opposed to the LLMs at least feasible to run using a CPU only. However, as shown in Table 4.3, this had quite a drastic impact on the indexing performance.

Had the open source models been used for retrieval, the performance difference would likely not have been as big. As shown in Table 4.4 the difference in computation time between *GPT-4* and *SFR-Embedding-Mistra* is much smaller for smaller documents. Computing the embedding of a user query, which is what is done during retrieval, equates to computing the embedding for a very small document.

*Massive Text Embedding Benchmark (MTEB) Leaderboard on Huggingface (accessed on May 1, 2024)

Prompt Length (No. characters)	0-256	257-512	513-1024	1025-2048	2049-4096	4097-8192
Model Name						
Salesforce/SFR-Embedding-Mistral	14.24s	17.48s	25.74s	47.13s	101.43s	150.43s
openai/text-embedding-3-large	**0.0**s	**0.02**s	**0.02**s	**0.06**s	**0.02**s	**0.03**s

Table 4.4: Average Total Response Time per Embedding Function and Prompt Length

4.2 The impact of different LLM models on the speed, accuracy and reliability of responses

This section will present and analyse the gathered data on user preference and technological efficacy of different tools and technologies such as different RAG toolchains and LLM, as outlined in section 1.4.

4.2.1 Thumbs up/Thumbs down responses to FAQ questions

After each response to a question that was selected from the frequently asked questions (FAQs) that were shown before a user had sent any messages, as shown in Figure 3.7, the user was presented with an optional binary thumbs up/down vote on the quality of the response. Both had a tooltip that said *"This was a good response"* and *"This was a bad response"* respectively. Table 4.5 shows how many users answered this question. Figure 4.15 shows how the participants in the study voted.

Question	**Possible Answers**	**No. Answers**
Was this a good reply?	Thumbs up, Thumbs down	78

Table 4.5: Questions asked to participants after receiving a response to one of the FAQ questions.

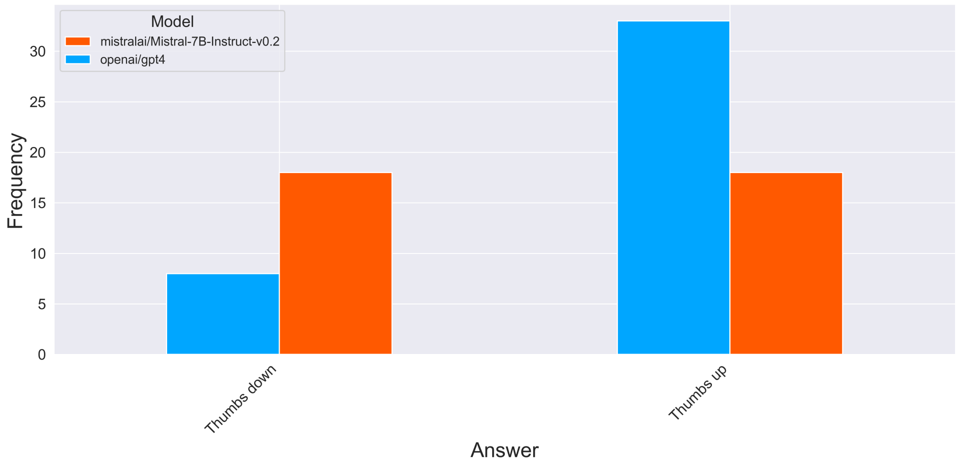

Figure 4.15: The number of answers to each answer for the question *"Was this a good reply?"*

Generally, it can be observed that users had a positive response to the replies produced to the FAQs. Almost twice as many positive answers were recorded as negative responses.

Looking at the breakdown per model, the open-source alternative *Mistral-7B-Instruct-v0.2*, had almost the same amount of positive and negative responses, which can be seen in Figure 4.15. The proprietary model *GPT-4* had a significantly higher proportion of positive responses, which indicates a generally more favourable reception from the participants in the study.

4.2.2 Survey questions injected into the chat

The software written for this thesis was, as discussed in sections 3.3.1.4 and 3.9.9, and shown in Figure 3.4, designed to gather user feedback by injecting questions in the chat at certain triggers. The questions inserted after receiving the first response in chat number 2, 4 and 6 of each session can be seen in Table 4.6. The questions inserted after receiving the first response in chat number 8 can be seen in Table 4.6.

Unfortunately, few of the users who actually used the system answered these questions. As Table 4.6 and Table 4.6 shows, many users answer the first set of questions, but very few used the system enough to get the questions inserted into their 8th chat, and those who did use it that much - did not answer it.

Question	Possible Answers	No. Answers
The reply from the bot was useful to me	Strongly agree, Agree, Neither agree nor disagree, Disagree, Strongly disagree	28
How accurate did you find the answers from the bot	Very accurate, Somewhat accurate, Neither accurate nor inaccurate, Somewhat inaccurate, Very inaccurate	26
How would you rate the speed of the bot's reply?	Very quick, Quick, Moderate, Slow, Very slow	31

Table 4.6: Questions asked to all study participants after receiving the first response in chat number 2, 4, 6.

Question	Answers	No. Answers
Overall, the information the bot provided to me has been useful	Strongly agree, Agree, Neither agree nor disagree, Disagree, Strongly disagree	0
Overall, how effectively has the bot been able to answer your questions?	Extremely effectively, Very effectively, Moderately effectively, Slightly effectively, Not at all effectively	0
Overall, the answers from the bot have been correct	Strongly agree, Agree, Neither agree nor disagree, Disagree, Strongly disagree	0
Overall, the answers from the bot contained all the information I needed	Strongly agree, Agree, Neither agree nor disagree, Disagree, Strongly disagree	0
How would you compare the ease of use of the bot with retrieving information from the canvas room yourself?	Much easier, Easier, Neither easier nor more difficult, More difficult, Much more difficult	0
How would you compare the time it takes to ask the bot about the canvas room with retrieving information from canvas yourself?	Much faster, Somewhat faster, About the same, Somewhat slower, Much slower	0

Table 4.7: Questions asked to all study participants after receiving the first response in chat number 8.

Since the number of answers weren't many, the study couldn't sample between more than two different configurations. The students did not know which configuration they got sampled, nor that there existed different configurations. These two configurations used different LLMs, but shared the same retrieval strategy (vector search) and embedding function (*openai/text-embedding-3-large*). In figures 4.16, 4.17 and 4.18 how users that were assigned the different LLMs answered can be seen.

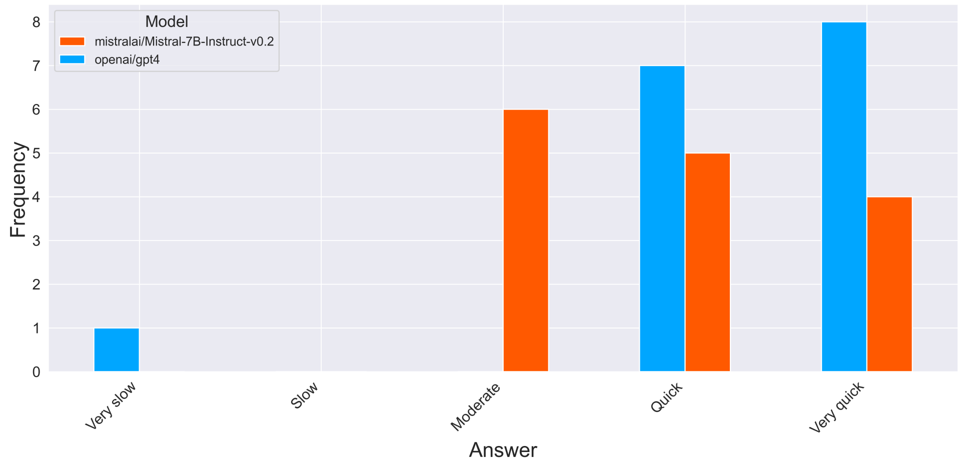

Figure 4.16: The number of answers to each answer for the question *"How would you rate the speed of the bot's reply?"*

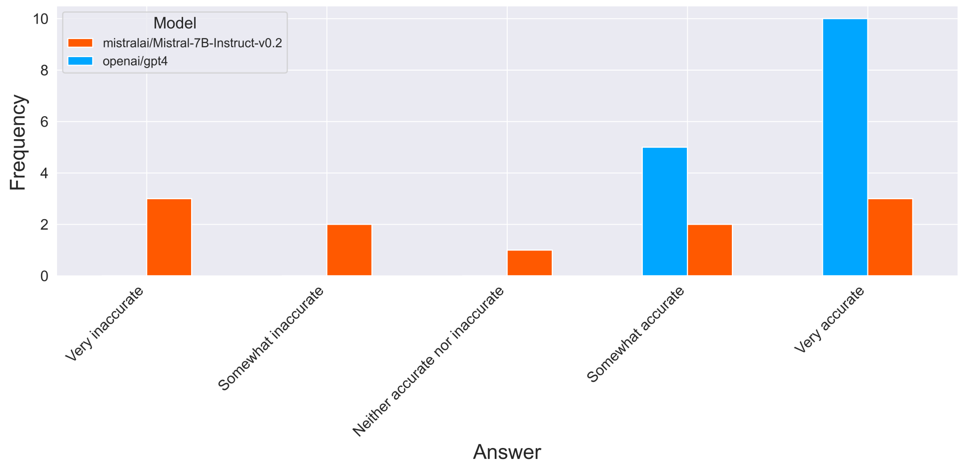

Figure 4.17: The number of answers to each answer for the question *"How accurate did you find the answers from the bot"*

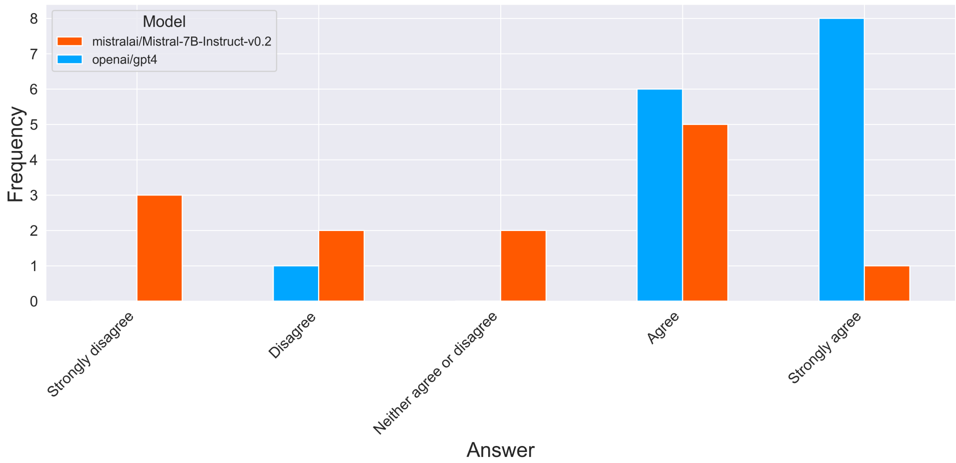

Figure 4.18: The number of answers to each answer for the question *"The reply from the bot was useful to me"*

4.3 Qualitative analysis of free-text answers

This section will present an analysis of the free text answers users have provided in the forms that have been presented in the participating courses. The complete answers to these forms can be found in Appendix B, Appendix C and Appendix D. With these answers there is no way to correlate the answers with which chat configuration they had used, such as what LLM had been used.

4.3.1 The form submitted in MG2040

4.3.1.1 Can you describe a situation where the chatbot was particularly helpful or fell short of your expectations?

The answers to this question show that users found the chatbot useful when asking questions about specific assignments, such as the Assembly project. One student say that it was surprisingly good at helping them with calculations for the project. Other used it to find certain topics and modules in the courseroom and thought the chatbot helped them to find it quickly.

There were also instances where the chatbot wasn't particularly helpful. One student said it couldn't provide the date of the exam. Others said its answers were too generic.

4.3.1.2 Which type of questions would you ask the chatbot as opposed to the teacher or teaching assistants?

The students used it to ask questions they deemed unnecessary to ask the course instructors directly. This could be questions about deadlines, number of lectures or other logistical details. Additionally, users mentioned asking the chatbot for detailed explanations with examples and for help with understanding course material. This could reportedly be late at night when course administrators aren't available.

Some users also prefer the chatbot for questions that might be too minor or unrecognised by teaching assistants. These could be questions that does not require the expertise of a teacher or TA.

4.3.1.3 Has using the chatbot changed the way you access information for your courses? If so, how?

The chatbot has altered the course room experience for some students. While some students reported minimal change, either because of the chatbot's novelty or lack of interest, others found it helped them to access information quicker. For instance, one student reported they asked the chatbot questions instead of searching through course room documents manually. Users also mentioned that the chatbot provided good explanations of content.

4.3.1.4 What has been your overall experience using the bot for course-related queries?

Overall students who tried the chatbot reported positive, and very positive, responses. One student reported it was surprisingly effective. Others reported that it was fun and easy to use. Although some responses are seemingly very generic, the general consensus seems to be that the chatbot has been a beneficial addition to their study resources.

4.3.1.5 Why have you not used the chatbot?

The students who began the form by indicating they had not tried the chatbot got the opportunity to answer the question why they had not tried it. Most users reported that they had not had time to try it, or it was not a priority of theirs to test it. One user expressed a lack of perceived necessity, seemingly satisfied with the current mechanisms of navigating the course material.

One user reported that the reason they had not tried the chatbot was because it was monitored by the teachers of the course. This indicates that some students might feel uneasy using this technology, because they are not sure what they are allowed to ask the chatbot.

4.3.2 The form submitted in LD1000

4.3.2.1 What was your experience using the chatbot for course-related questions?

Users found the chatbot to be helpful. This was especially true for the course-related queries. Some users noted that the chatbot was very detailed in its answers, others thought it provided very high-level answers. One user noted that the bot included references to where more info could be found, which that user thought was very helpful. Some users expressed that they were surprised how well it worked. Overall the responses suggests that the chatbot was generally well-received.

4.3.2.2 Were there situations where the chatbot was particularly helpful or did not meet your expectations?

Most responses recorded a situation they thought were particularly helpful. No user used this question to report an event they thought the bot did not meet their expectations. Users found the chatbot helpful when it provided detailed information, including links to content in the course room. Some users reported that they found it useful to be able to ask follow-up questions, with more tailored questions depending on the answer to their first questions.

4.3.2.3 Are there questions you prefer to ask the chatbot instead of a teacher or teaching assistant? Which ones?

Answers here generally fell into two categories of seemingly similar size. One category reported that they appreciate the ability to ask the bot simpler questions, perhaps about course logistics, when the teacher isn't online or available. Some users prefer the immediacy and convenience of asking a chatbot. Some still noted though that they generally prefer the human connection of asking a teacher or TA.

The second category was users who reported they don't prefer asking any questions to a chatbot over a teacher. The reason here was generally that they preferred the "human-connection" of asking a human teacher. Where one suggested there was a trust issue with asking a chatbot over a human.

4.3.2.4 Do you think using the chatbot can change how you access information in a course? How?

The responses to this question generally indicate students think it would mostly change how they access information in courses with larger canvas rooms. That is canvas rooms with lots of pages and difficult to understand navigation. Some reported that they generally would prefer better structured course rooms, but when that's not possible the chatbot could be a nice complement.

Some users didn't think the chatbot would change how they accessed information. This was either due to not having used it much in this course, or because they thought they would prefer talking to a human teacher.

4.3.2.5 What do you think about using AI tools in your studies? Do you use any tools today? Which ones, and for what?

A minority of the responses reported that they used AI tools in their academic studies. Some reported that they used it for programming assignments. Others said that they use it for explanations or clarifications in texts they've written. Most did report that they don't use it though. Some reasons for this were that they hadn't found it useful, preferred to do stuff themselves because it was more creative and it could be considered cheating.

4.3.3 The form submitted in LD1006

4.3.3.1 What was your experience of using the chatbot for course-related questions?

The responses to this question were mixed. Some users seemed surprised at how well it worked, some thought the answers were too vague to be useful. Some users reported that the answers were inaccurate, one user provided an example where the chatbot had said zoom meetings were recorded and published on the canvas page, something they claimed was inaccurate. Many answers were positive, where users claimed they've received accurate information.

4.3.3.2 Were there any situations where the chatbot was particularly helpful or where it fell short of your expectations?

Generally responses here fall into two categories. The first category is users who are satisfied with the answers and think the chatbot answered their questions satisfactorily. Some provide extra information of a particular instance where the chatbot answered a specific question. The second category is unsatisfied users. The reason they report they aren't satisfied varies, some report that the bot failed at answering a specific question. One user reported that the chatbot failed at providing references to the course literature.

There is also a small number of users who can't recall a specific instance where the bot either exceeded or did not meet their expectations.

4.3.3.3 Are there questions you would rather ask the chatbot than a teacher or teaching assistant? Which ones?

Responses to this question mainly indicate that users can't think of any given question that they would prefer to ask. Most users simply report that they can't think of any such questions. Some answers indicate that with more usage they might think of questions they would prefer to ask.

Some users answer more generally that they either weren't exactly sure who to ask about a certain question, and in scenarios like that, the chatbot might be preferable. Similarly, when either the teacher isn't available or the question is very simple, their answer indicate that they prefer a chatbot

4.3.3.4 Do you think the use of the chatbot could change how you access information in a course? How?

The responses indicate that many users are sceptical of the chatbot having a material impact on how they access course information. Some users report that it might, some are ambiguous. Those who think that it might change how they access course information mention that it might be useful for certain particular tasks, such as summarisation. Others think that if the chatbot's intelligence generally improves, the bot will change how they access information.

Those who think that it won't change how they access information, sometimes reference they are new to the technology and aren't sure how to use it. Most responses that indicate the user doesn't think that it will change how they access course information, don't elaborate on why.

4.3.3.5 How do you think about the use of AI tools by both yourself and your students? Do you and/or your students use any tools today? Which ones, for what?

This question prompted the longest answers, many responded with very detailed answers indicating this is a question many respondents feel strongly about. The answers indicated a broad range of attitudes towards the use of AI tools in the educational setting. Some users are positive about the potential of AI tools and their ability to enhance student learning. Some respondents included stories where their students had used AI for student-to-student discussions, providing quick information, and aiding in planning and breaking down assignments. Those who are positive generally still stress the importance of educating both students and educators about AI risks, and how to use it effectively. Some mention that AI will be more effective in certain subjects.

The users who expressed caution about using AI within education often had stories in which AI had been used with either little success or very poor outcome. One respondent mentioned students had stopped using it for solving their word problems in their class because its results were simply too bad. Many respondents mentioned ChatGPTs, and other AIs like it, poor abilities in solving maths problems, as hinders to being used in education.

Some responses indicated that the respondent felt generally unprepared to use AI. One respondent admits to not be using AI tools due to their own lack of knowledge. They expressed a need to learn more before incorporating them into teaching their own teaching.

Many respondents raised concerns about authenticity and plagiarism. They want the ability to distinguish between work created by an AI and their student's own work. Another common theme was the importance of critical thinking, where one response emphasised the need to teach students how to critically read and evaluate AI-generated content to ensure they can understand what a model generates is high quality, and what is lower quality.

Chapter 5

Discussion

This chapter will discuss the key findings from the research carried out in this thesis. It will evaluate the implications of the findings and compare it to existing literature. The discussion is structured to provide analysis of the results. It will also explore why some questions laid out in the introduction section have been left unanswered. Finally the chapter will include a brief discussion about the role of AI assistants in the broader context of AI in the educational setting.

5.1 Feasibility of Open Source Technologies

The study, and the software constructed to execute its experiments, show that it is feasible to build and deploy an AI assistant using open source technologies. The most popular models to implement AI assistants are commonly provided under proprietary licences and hidden behind pay-per-use APIs. Examples of this are the models built and sold by OpenAI.

The results laid out in chapter 3 and chapter 4 shows that an open source alternative is feasible from an implementation standpoint. The open source LLM and embedding functions used in the study proved functional and provided comparable performance to proprietary solutions with respect to operational metrics such as response time and query generation time. One learning from operating the software constructed for the study is that the embedding function should probably be operated with a dedicated GPU to improve performance when computing embeddings of larger docs, such as during indexing.

5.2 User Engagement and Satisfaction

The study showed steady engagement from students across multiple different courses. The courses were all fairly different from each other. Some courses were self-study courses, without any lectures, some had regular lectures. Some courses had many participating students, some courses had as little as 30 students in the course.

There was a clear indication that the proprietary models such as OpenAI's GPT-4 received more favourable feedback compared to its open source counterparts. Unfortunately, due to the volume of participating courses and the number of chats that were held, the decision was made to not test other variables, such as different RAG techniques, or embedding functions.

5.3 Why AI assistants are helpful to students

Chapter 4 included section 4.3 which analysed the answers collected from students in three courses who had tried the chatbot. The responses to questions distributed in these courses provided valuable insights into why students responded like they did to the questions in section 4.2.2.

The first conclusion that can be drawn from the responses to the form is the benefit students feel of being able to quickly access information at any time. Instead of having to navigate extensive course materials, students can quickly get answers to their questions by simply asking the AI assistant in the course room. It is easy to forget that students are enrolled in multiple courses at the same time, each with a different course room utilising a different course room layout. Even if an individual course room is well-designed, the simple fact that another course room is poorly designed, can make navigation challenging.

Being able to ask an assistant anything, at any time, was particularly useful for logistical details such as deadlines, lecture schedules, and the links to course material. This was especially mentioned by the students in the *MG2040 Assembly Technology* course, however, these themes were present in the answers from students in all three courses which distributed forms.

Another thing students appreciated was being able to use the chatbot to understand course content. For instance, in the *LD1000* course, users noted that the AI assistant provided very extensive answers and good explanations.

One core component was the integration of citations. Citations aren't necessarily required when constructing chatbot-applications. However, many students, across all courses, often mentioned the importance of understanding the source of the assistants responses for being able to trust the answer. The assistant's ability to both provide comprehensive answers, and links to follow-up information, was cited as a key reason by some students, as why they had felt the bot was useful.

Some students even reported that they preferred asking the assistant some questions over asking the teachers and TAs. This was primarily due to the fact that the assistant was available all hours of the day, and could provide quicker responses. Also, some students reported that they felt their questions could be too simple or trivial for asking a teacher. This is an interesting insight because this suggests there are questions that students aren't asking teachers today. Ensuring that students are comfortable asking questions about the course and its contents is key for ensuring that students learn efficiently.

5.4 Methodological Improvements

The software designed for the study aimed to test various tools and techniques commonly used when constructing AI assistants, in alignment with the goals outlined in 1.4. However, shortly after the study's launch, it became evident that the number of participating courses - and consequently, the number of participating students - was insufficient to provide statistically significant results for more than one feature.

The number of participating courses was a function of when the custom software, developed for the study, was finished. Several course administrators and teachers at KTH were contacted and showed interest in participating in the study. The most common reason for not participating was that the course did not run during the study period, i.e. the course responsible had courses in P1, P2 or P3, not P4. The second most common reason was that the teacher held courses that had already started, and did not want to make changes to ongoing courses.

Numerous tools and techniques, which were time-consuming to implement, were ultimately not enabled in the experiments. The study would have been more successful if it had started earlier, allowing more time to accumulate students and participating courses. Furthermore, the courses that did participate had already been live for at least a few weeks, sometimes more than a month. It is likely that students have more questions that would've been appropriate for a chatbot at the beginning of a course. Therefore, the study would have also benefited from starting at the beginning of a study period, rather than in the middle of P4.

The feedback mechanism was designed to ask students questions at controlled time intervals. As detailed in chapter 3, this approach aimed to account for the students' familiarity with the AI assistant. The length of time a student used the tool could influence their responses. For this reason, the study was structured to introduce questions at specific intervals, which were determined after a certain number of chat sessions.

However, as can be seen in figures 4.8 and 4.9, many users sent a substantial number of messages, but fewer users had more than two chat sessions. Therefore, a better measure of a user's familiarity with the assistant might have been the number of messages sent. This alternative method would have prompted more users to provide feedback at more consistent intervals.

5.5 The tools, models and technologies that were selected for the experiments

It's worth noting that the comparison between the open and closed-source models used in this experiment is not entirely fair. *Mistral-7B-Instruct-v0.2* has 7 billion parameters. OpenAI discloses less about their models; however, *GPT-3* has a confirmed 175 billion parameters. Less is known about *GPT-4*, which is rumoured to have a SMoE architecture, likely with at least as many, if not more, parameters than *GPT-3*. Even though the number of parameters isn't the best metric for determining intelligence in a model, it is one useful metric for comparing models. Had it been operationally and financially feasible to run a larger, more similar model, such as *Mixtral-8x7B-Instruct-v0.1*, that would've made for a better comparison.

It could also be argued that the *GPT-3* model could have been chosen, since that would've made for a fairer comparison. Since *GPT-4* is the current best model, and one that likely would have been used implementing tools like this in real world applications, that model made sense to test with respect to measuring the general acceptance of AI assistants among students. Had there been more time, or more money to fund more expensive servers to run the larger open source models, the experiment would have been executed with different models. The same cannot be said for the embedding functions though.

Even though the embedding functions incorporated in the study were never tested for retrieval purposes, due to the number of participating courses, the models selected should be easier to compare. *Mistral-7B-Instruct-v0.2* and *GPT-4* score very differently on the available benchmarks. The embedding functions chosen, Salesforce's *SFR-Embedding-Mistral* and OpenAI's *text-*

embedding-3-large, rank very similarly to each other on the MTEB.

5.6 Prompts and Instructions Aren't Portable Across Models

Just as the study was launched, Meta released the third iteration of their LLaMA family of models, LLaMA 3. This included *LLaMA 3 8B*, an eight billion parameter model. A model of this size was feasible to operate and scored significantly higher than *Mistral-7B-Instruct-v0.2* on most benchmarks. Since both are open source, *LLaMA 3 8B* obviously made for a better candidate for testing open source LLM.

The infrastructure to download and run the new model is very flexible and stable. The software was quickly adapted to switch the study from Mistral's model to Meta's latest model. However, it quickly became evident that the prompts used in the application were incompatible. As explained in section 2.4, prompts are a key component of getting the LLM to perform as desired. In this study's software, this primarily involved hosting a chat with the user and some internal logic for constructing queries and indexing/retrieving documents. These prompts weren't trivially ported to work well with LLaMA 3.

The prompts were designed and tested with *Mistral-7B-Instruct-v0.2* and *GPT-4* during the initial development of the study's software. Quickly adapting these prompts to also work with *LLaMA 3 8B* wasn't possible. One key takeaway for when constructing similar studies is that model selection must be done early and continuously monitored to ensure accurate test results. In this study, *GPT-4* and *Mistral-7B-Instruct-v0.2* used the same prompts. One could even argue that this should be a variable controlled for to ensure both models have the best available and most adapted instructions for them to use.

5.7 Ethical and privacy concerns

The deployment of an AI tool like the assistant developed for this raises several ethical concerns, primarily regarding privacy and the trust feel for AI tools. Some answers in the forms provided in this course show that many students hold great concerns about AI tools and assistants, and their role within their educational environment.

An important concern that was raised in responses from students was the potential monitoring of the questions and interactions with the chatbot by the teachers. As was raised by one student in the forms, they explicitly stated *"[the chat] is monitored by the teacher"* as the reason they hadn't used the chat bot. Even though the assistant did offer a way for teachers to view the anonymous chats students had with the assistant, students weren't told this fact. It's noteworthy that a student simply presumed this to be true.

When presenting the assistant to teachers interested in participating in the study it was raised as a requirement from all of them, that there had to be a way for them to view the conversations. For instance, if the chatbot had answered a question incorrectly, and a student referred to this answer, the teacher needed the ability to verify what the chatbot had said. Therefore, this seems like a pressing issue to work on before deploying chatbots, like the one in this course, more broadly across schools and universities.

Before using the assistant in this course there was a very clear and unambiguous ethics note. The note featured a section about anonymity. The user would remain completely anonymous when interacting with the chatbot. Even though this was the case some users still didn't trust the AI assistant was anonymous.

In a classroom, with a teacher or TA, there is no chance for anonymity, at least without using anonymising tools, such as online polls. AI assistants, like the one constructed in this study, present a huge opportunity for allowing students to anonymously ask teachers questions. Remaining anonymous can be important for several reasons. Students can feel questions they ask might influence their grades or be too embarrassed to ask a question they think makes them seem stupid in front of their peers. Even though this bot, according to some users, failed to provide an environment of anonymity. This remains an important feature to explore further in future work.

At some point, it might be useful for the user to not remain anonymous. Students might want to forward their question to a teacher if they are suspicious the answer generated by the assistant might be inaccurate. It could also be the case that the bot will need personal information to respond to the students request.

The bot in this course only had access to information available to all students. Developing a version with access to student specific information, such as their past assignments or grades, puts increased scrutiny on the correctness of the systems. Allowing students to grant access to their data for specific queries might be an interesting avenue to explore. For instance, if a student asks questions like *"how many points do I need on my next homework in order to meet the requirements for an A?"*, the chatbot could reply with an authorisation request from the student to view their past work. Mechanisms like this are key to explore, for improving student trust in their anonymity while enhancing the usefulness of the assistant.

Chapter 6

Conclusions and Future work

This chapter will provide conclusions from the research carried out in this thesis. It will reflect on the goals laid out in section 1.4 and what insights have been gained. It will describe the limitations to the result and discuss some future work.

6.1 Conclusions

One of the goals for this thesis was to understand the technological efficacy, i.e. speed and accuracy of various different tools and techniques commonly used when developing AI assistants. In addition to that, this thesis tried to understand what users preferred, various tools and the feasibility of operating LLMon-premises. Lastly, a goal was to understand how AI assistants impacted education. All of this was encapsulated by the research questions, *Which language model and which retrieval techniques do students prefer using?* and *Is it possible to deploy an AI-assistant using a completely open source toolchain?*. My hypothesis was that the closed source alternatives would be better, however it would be feasible to build a completely open source and self-hosted AI assistant too.

Generally, the hypothesis was correct. The results laid out in chapter 4, and section 4.2.2 specifically, showed that students preferred the model provided by OpenAI to the Mistral model. Even though that comparison isn't very fair model-to-model, the fact that self-hosting models that are as large as those available by for profit vendors, means that building an assistant on proprietary models will yield a better assistant. However, the results show generally favourable opinions from the users who had to use the open source model.

Due to the size of the experiment, there weren't enough participants to test all the tools and techniques initially intended. The software constructed for the experiments was designed for testing more tools and models, such as some open source embedding functions. However, given these weren't used in real chats, no conclusions were drawn regarding their effectiveness or effect on student satisfaction. For the same reason, no data was collected on how traditional search techniques, such as fulltext search, affected the metrics collected in the study.

The perceived speed, as reported by the users, was just slightly slower for the open source model used, as can be seen in **??**. This was also backed up by the recorded time each response took in Figure 4.10. The only measurement of accuracy in this study was the perceived accuracy by the users. This is a vert brute metric, however, it did produce a clear result that users perceived *GPT-4* to produce more accurate results than the Mistrals model. Again, this was fairly inline with expectations. Larger open source models might produce even more accurate results. However, a model as capable as *Mistral-7B-Instruct-v0.2*, is not capable enough of building an AI assistant that users overwhelmingly think produce accurate results. The same conclusion can be drawn for the perceived usefulness of the assistants built with the different models.

Finally, the research conducted in this thesis did produce some insights on what impact chatbots and AI assistants could have on education. Students appreciated the quick and easy access to information. AI assistants could definitely improve student-teacher communications. This doesn't mean that all student inquiries should be answered by a chatbot. Students prefer speaking to their actual teacher, however, a chatbot is a useful complement. While implementing such chatbots, the research in this thesis shows the technology selection is important. It is very important the information the bot has access to is accurate. While privacy is paramount for students, it's still crucial teachers has insight into what the chatbot is telling students.

So in conclusion, the first research question that asked if it was feasible to build and deploy an AI-assistant using a completely open source toolchain can be considered answered. It was feasible, and was done in this thesis. Also, the latter that asked which language models and tools users of the chatbot would prefer was partially answered. The experiments showed conclusive answers in terms of what language models users preferred, other tools and techniques weren't evaluated due to the limited size of participants in the study.

6.2 Limitations

The experiments carried out in this thesis were partially able to answer the research questions laid out in chapter 1. The study would have benefited from more courses that were included in the study. The effort of soliciting courses to join the study was only started in late march, early april. This meant many courses in P4 had already started. This limited the number of courses that were interesting in joining the study, many hesitated changing the format of already started courses. In addition, the courses that did join the study had entered their final half, or final weeks in some cases. This meant that the students probably had already internalised much of the course logistics. Similar studies would with benefit be planned further in advance, ensuring that the experiment is launched just before or in the beginning of a new course. Courses could even be designed around using the chatbot.

Designing studies like the one in this thesis, where large custom software is designed and implemented, would benefit from a more incremental approach. The process for the research in this thesis involved very time consuming software development in the beginning, only to be deployed with too little time to test everything that was included in the software. Therefore, a better approach would have been getting a working assistant, with less features deployed faster. Then iteratively, as there is time left in the study, work on more features. For instance, the experiments never had time to run with post processing of documents enabled, which was a feature that took several weeks to design, implement and test.

6.3 Future work

There is some obvious future work that was left undone within this thesis. For instance, running the same experiments with different tools and models. This includes testing open source embedding functions as a retrieval technique, and evaluating those similarly to what was done for ?? in this thesis, both against proprietary embedding functions and other retrieval techniques such as full text search and TF-IDF.

In addition to this, exploring how chatbots can be granted access to student specific content. Such as their personalised schedule, past assignments and other courses. There are a myriad of other logistical courses students might want to get help with. These include what courses they are eligible for, what they are required to take etc. In all of these cases a chatbot might be a useful tool to deploy. Therefore, future research should be done to investigate how assistants can be given access to this data safely, whilst maintaining user trust as discussed in section 5.7.

6.4 Reflections on sustainability

AI is not a new field of research, and it has been integrated into products and services for years. However, in the last few years their immediate usefulness has become more apparent, especially with the release of tools such as ChatGPT. It is clear the new LLMs are very capable and useful. The technology is still very nascent. In the coming years and decades integrating these models into real world applications and business processes will be a long process.

Training machine learning models has traditionally been very compute-intensive, often requiring accelerated compute devices such as GPUs. Computation requires electricity, which depending on the electricity source, means an increased environmental footprint. By extension this means compute intensive applications, such as training models, have an impact on the environment. What's new with the development of LLMs, is that inference, the activity of merely running the models, is also very compute intensive.

The software constructed in this study supported adding server nodes that could run a LLM on an attached GPU. During most of the study however, only one server node was used, since the number of students rarely peaked above more than one or two at a time. If the chatbot were to be enabled in more course rooms however, the demands on the available amount of compute-capacity would increase.

The fact that operating applications built on-top of LLMs require large amounts of compute capacity, leads to interesting discussions around the Sustainable Development Goal (SDG) goals laid out by the United Nations. In the case of the research in this thesis, SDG numbers four and thirteen are the most relevant. Goal number four tries to ensure high quality education, while goal thirteen tries to combat climate change and its effects. The research in this thesis has shown that AI-assistants can increase the quality of education and improve the student learning experience. However, since this requires running LLM, which has an environmental impact, this results in a trade-off between education and environmental impact.

It's not clear that LLM inference will always have a high environmental impact. New inference technologies such as Groq [4] might result in faster inference with lower energy usage. Achieving energy efficiency in inference would essentially eliminate the tradeoff between the two SDG goals, although this remains to be proven.

Operating a LLM on-premise isn't always necessary though. However, running the language model yourself offers major privacy benefits. In the case of the application built in this thesis, KTH may need to keep all student-data on their own servers, which could also be the case in other applications in completely different domains, either due to legal or contractual obligations. Besides privacy, performance can also justify on-premise deployments. Having dedicated models available ensures that the performance of applications relying on the LLM isn't impacted by the load on third party vendors, such as OpenAI.

By running the model on-premise, this has a higher likelihood of underutilised hardware. Using energy whilst not executing prompts, keeping a model in memory whilst being idle. Using a third party vendor, such as OpenAI or Groq, leads to the pooling of resources between AI applications. This could lead to a higher utilisation of the existing hardware, and reduce the environmental impact. What becomes the de facto standard when building LLM application in the future, remains to be seen.

References

[1] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the Middle: How Language Models Use Long Contexts," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, Feb. 2024. [Online]. Available: https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00638/119630/Lost-in-the-Middle-How-Language-Models-Use-Long [Pages 2 and 14.]

[2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., Dec. 2020, pp. 9459–9474. [Pages 2, 23, 24, and 25.]

[3] G. Team, M. Reid, N. Savinov, D. Teplyashin, Dmitry, Lepikhin, T. Lillicrap, J.-b. Alayrac, R. Soricut, and A. Lazaridou, "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context," Apr. 2024, arXiv:2403.05530 [cs]. [Online]. Available: http://arxiv.org/abs/2403.05530 [Pages 2, 14, 18, and 24.]

[4] D. Abts, G. Kimmell, A. Ling, J. Kim, M. Boyd, A. Bitar, S. Parmar, I. Ahmed, R. DiCecco, D. Han, J. Thompson, M. Bye, J. Hwang, J. Fowers, P. Lillian, A. Murthy, E. Mehtabuddin, C. Tekur, T. Sohmers, K. Kang, S. Maresh, and J. Ross, "A software-defined tensor streaming multiprocessor for large-scale machine learning," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*. New York New York: ACM, Jun. 2022, pp. 567–580. [Online]. Available: https://dl.acm.org/doi/10.1145/3470496.3527405 [Pages 2 and 92.]

[5] L. Basyal and M. Sanghvi, "Text Summarization Using Large Language Models: A Comparative Study of MPT-7b-instruct, Falcon-7b-instruct,

and OpenAI Chat-GPT Models," Oct. 2023, arXiv:2310.10449 [cs]. [Online]. Available: http://arxiv.org/abs/2310.10449 [Page 2.]

[6] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. E. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. J. Lowe, "Training language models to follow instructions with human feedback," *Neural Information Processing Systems*, Mar. 2022. [Pages 2 and 14.]

[7] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M.-A. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mixtral of Experts," Jan. 2024, arXiv:2401.04088 [cs]. [Online]. Available: http://arxiv.org/abs/2401.04088 [Pages 3, 14, and 18.]

[8] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring Massive Multitask Language Understanding," *International Conference on Learning Representations*, Sep. 2020. [Pages 3, 17, and 26.]

[9] Y. Bisk, R. Zellers, R. Le Bras, J. Gao, and Y. Choi, "PIQA: Reasoning about Physical Commonsense in Natural Language," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, Apr. 2020, pp. 7432–7439, iSSN: 2374-3468, 2159-5399 Issue: 05 Journal Abbreviation: AAAI. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6239 [Pages 3 and 17.]

[10] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training Verifiers to Solve Math Word Problems," Nov. 2021, arXiv:2110.14168 [cs]. [Online]. Available: http://arxiv.org/abs/2110.14168 [Pages 3 and 17.]

[11] S. Siriwardhana, R. Weerasekera, E. Wen, T. Kaluarachchi, R. Rana, and S. Nanayakkara, "Improving the Domain Adaptation of Retrieval Augmented Generation (RAG) Models for Open Domain Question Answering," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 1–17, Jan. 2023. [Online]. Available: https://doi.org/10.1162/tacl_a_00530 [Pages 3 and 27.]

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/323533a0 [Page 9.]

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, conference Name: Proceedings of the IEEE. [Online]. Available: https://ieeexplore.ieee.org/document/726791 [Page 9.]

[14] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. [Page 10.]

[15] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://aclanthology.org/D14-1179 [Page 10.]

[16] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, Dec. 2014, pp. 3104–3112. [Page 11.]

[17] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *CoRR*, Sep. 2014. [Pages 11 and 12.]

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html [Pages 11 and 12.]

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,"

in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423 [Pages 11 and 25.]

[20] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," Oct. 2016, arXiv:1609.08144 [cs]. [Online]. Available: http://arxiv.org/abs/1609.08144 [Page 12.]

[21] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov. 2012, conference Name: IEEE Signal Processing Magazine. [Online]. Available: https://ieeexplore.ieee.org/document/6296526 [Page 12.]

[22] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-Shot Text-to-Image Generation," in *Proceedings of the 38th International Conference on Machine Learning*. PMLR, Jul. 2021, pp. 8821–8831, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v139/ramesh21a.html [Page 12.]

[23] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html [Pages 12, 13, and 24.]

[24] OpenAI, "Video generation models as world simulators," Mar. 2024. [Online]. Available: https://openai.com/index/video-generation-models-as-world-simulators [Page 12.]

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *Advances in Neural Information Processing Systems*, vol. 3, Jun. 2014. [Page 12.]

[26] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7B," Oct. 2023, arXiv:2310.06825 [cs]. [Online]. Available: http://arxiv.org/abs/2310.06825 [Page 13.]

[27] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebron, and S. Sanghai, "GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 4895–4901. [Online]. Available: https://aclanthology.org/2023.emnlp-main.298 [Page 13.]

[28] A. Roy, M. Saffar, A. Vaswani, and D. Grangier, "Efficient Content-Based Sparse Attention with Routing Transformers," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 53–68, 2021, place: Cambridge, MA Publisher: MIT Press. [Online]. Available: https://aclanthology.org/2021.tacl-1.4 [Page 13.]

[29] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, and K. Millican, "Gemini: A Family of Highly Capable Multimodal Models," Apr. 2024, arXiv:2312.11805 [cs]. [Online]. Available: http://arxiv.org/abs/2312.11805 [Page 14.]

[30] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, and J. Love, "Gemma: Open Models Based on Gemini Research and Technology," Apr. 2024, arXiv:2403.08295 [cs]. [Online]. Available: http://arxiv.org/abs/2403.08295 [Page 14.]

[31] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez,

A. Joulin, E. Grave, and G. Lample, “LLaMA: Open and Efficient Foundation Language Models,” Feb. 2023, arXiv:2302.13971 [cs]. [Online]. Available: http://arxiv.org/abs/2302.13971 [Pages 14 and 18.]

[32] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, “Llama 2: Open Foundation and Fine-Tuned Chat Models,” Jul. 2023, arXiv:2307.09288 [cs]. [Online]. Available: http://arxiv.org/abs/2307.09288 [Page 15.]

[33] S. Kathiriya, M. Mullapudi, and A. Shende, “The Power of Prompt Engineering: Refining Human -AI Interaction with Large Language Models in The Field of Engineering,” *International Journal of Science and Research (IJSR)*, vol. 12, Nov. 2023. [Pages 15 and 16.]

[34] B. Chen, Z. Zhang, N. Langrené, and S. Zhu, “Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review,” Oct. 2023, arXiv:2310.14735 [cs]. [Online]. Available: http://arxiv.org/abs/2310.14735 [Pages 15 and 16.]

[35] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, and C. Sutton, “Program Synthesis with Large Language Models,” Aug. 2021, arXiv:2108.07732 [cs]. [Online]. Available: http://arxiv.org/abs/2108.07732 [Page 17.]

[36] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse,

A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, "Evaluating Large Language Models Trained on Code," Jul. 2021, arXiv:2107.03374 [cs]. [Online]. Available: http://arxiv.org/abs/2107.03374 [Pages 17 and 18.]

[37] M. AI, "Introducing Meta Llama 3: The most capable openly available LLM to date," Apr. 2024. [Online]. Available: https://ai.meta.com/blog/meta-llama-3/ [Page 18.]

[38] G. Pant and P. Srinivasan, "Crawling the Web," *Web Dynamics*, Jul. 2003. [Page 18.]

[39] M. Najork, "Web Crawler Architecture," L. Liu and M. T. Özsu, Eds. Boston, MA: Springer US, 2009, pp. 3462–3465, book Title: Encyclopedia of Database Systems. [Online]. Available: http://link.springer.com/10.1007/978-0-387-39940-9_457 [Page 18.]

[40] M. Christopher D., R. Prabhakar, and S. Hinrich, *Introduction to Information Retrieval*. Cambridge University Press, 2008. [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/irbook.html [Pages 19 and 20.]

[41] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," Jan. 2013. [Online]. Available: https://www.semanticscholar.org/paper/Efficient-Estimation-of-Word-Representations-in-Mikolov-Chen/f6b51c8753a871dc94ff32152c00c01e94f90f09 [Pages 21 and 22.]

[42] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems*, vol. 26. Curran Associates, Inc., 2013. [Online]. Available: https://papers.nips.cc/paper_files/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html [Page 22.]

[43] J. Pennington, R. Socher, and C. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds. Doha, Qatar:

Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://aclanthology.org/D14-1162 [Page 22.]

[44] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "MTEB: Massive Text Embedding Benchmark," in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, A. Vlachos and I. Augenstein, Eds. Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 2014–2037. [Online]. Available: https://aclanthology.org/2023.eacl-main.148 [Page 22.]

[45] R. Meng, Y. Liu, S. Rayhan Joty, C. Xiong, Y. Zhou, and S. Yavuz, "SFR-Embedding-Mistral: Enhance Text Retrieval with Transfer Learning," Feb. 2024. [Online]. Available: https://blog.salesforceairesearch.com/sfr-embedded-mistral/ [Pages 22 and 69.]

[46] OpenAI, "New and improved embedding model," Dec. 2022. [Online]. Available: https://openai.com/index/new-and-improved-embedding-model [Page 22.]

[47] ——, "New embedding models and API updates," Jan. 2024. [Online]. Available: https://openai.com/index/new-embedding-models-and-api-updates [Pages 22 and 67.]

[48] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "REALM: retrieval-augmented language model pre-training," in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20, vol. 119. JMLR.org, Jul. 2020, pp. 3929–3938. [Page 23.]

[49] D. Sachan, M. Patwary, M. Shoeybi, N. Kant, W. Ping, W. L. Hamilton, and B. Catanzaro, "End-to-End Training of Neural Retrievers for Open-Domain Question Answering," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 6648–6662. [Online]. Available: https://aclanthology.org/2021.acl-long.519 [Page 24.]

[50] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*, third edition, global edition ed., ser. Prentice Hall series in artificial intelligence. Boston Columbus Indianapolis New York San Francisco Upper Saddle River Amsterdam Cape Town Dubai London Madrid

Milan Munich Paris Montreal Toronto Delhi Mexico City Sao Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo: Pearson, 2016. [Page 24.]

[51] J. Nielsen, "Chapter 2 - What Is Usability?" in *Usability Engineering*, J. Nielsen, Ed. San Diego: Morgan Kaufmann, Jan. 1993, pp. 23–48. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B978008052029250005X [Page 24.]

[52] J. Lewis and J. R., "IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use," *International Journal of Human-Computer Interaction*, vol. 7, p. 57, Feb. 1995. [Page 24.]

[53] T. Bickmore and J. Cassell, "Social Dialogue with Embodied Conversational Agents," Jan. 2005. [Page 24.]

[54] F. D. Davis, "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology," *MIS Quarterly*, vol. 13, no. 3, pp. 319–340, 1989, publisher: Management Information Systems Research Center, University of Minnesota. [Online]. Available: https://www.jstor.org/stable/249008 [Page 25.]

[55] A. Bhattacherjee, "Understanding Information Systems Continuance: An Expectation-Confirmation Model," *MIS Quarterly*, vol. 25, pp. 351–370, Sep. 2001. [Page 25.]

[56] I. Beltagy, K. Lo, and A. Cohan, "SciBERT: A Pretrained Language Model for Scientific Text," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3615–3620. [Online]. Available: https://aclanthology.org/D19-1371 [Page 25.]

[57] J. Hernández-Orallo, "Evaluation in artificial intelligence: from task-oriented to ability-oriented measurement," *Artificial Intelligence Review*, vol. 48, no. 3, pp. 397–447, Oct. 2017. [Online]. Available: https://doi.org/10.1007/s10462-016-9505-7 [Page 26.]

[58] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, J. Su, K. Duh, and X. Carreras, Eds. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. [Online]. Available: https://aclanthology.org/D16-1264 [Page 26.]

[59] K. Zhang, J. Wang, E. Hua, B. Qi, N. Ding, and B. Zhou, "CoGenesis: A Framework Collaborating Large and Small Language Models for Secure Context-Aware Instruction Following," Jun. 2024, arXiv:2403.03129 [cs]. [Online]. Available: http://arxiv.org/abs/2403.03129 [Page 26.]

[60] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei, "Text Embeddings by Weakly-Supervised Contrastive Pre-training," Feb. 2024, arXiv:2212.03533 [cs]. [Online]. Available: http://arxiv.org/abs/2212.03533 [Page 39.]

Appendix A

Source Code

The source code used to conduct the research in this thesis is available in its entirety at github.com/nattvara/kth-assistant. Furthermore, the code for this thesis and the notebooks used to produce the results is available at github.com/nattvara/DA231X.

Appendix B

All answers to the form submitted in MG2040

This includes all answers to the questions in the form submitted to the students in the course *MG2040 Assembly Technology 6.0 credits*. The form included logic, so only students who had used the bot got shown questions evaluating the bot. Student who answered they hadn't used the bot, got shown a single question asking them why they hadn't used it.

Form Submission No.	Answer
1	During Assembly project assignment I had some help with the calculations and surprisingly it returned commendable results.
3	To search certain topics from the modules.
4	While searching for the project.
5	Assisted me explaining the notes.
8	It couldn't answer when the exam is.
7	Mostly when I have doubts that I can't find in the canvas.
17	Easy knowledge questions.
19	When I need to find quickly information in the slide.
20	I only used it once for instructions on the line balancing. It gave me the information but it was still quite a generic answer and needed to look into the slides for full understanding.

Table B.1: Answers to the question: Can you describe a situation where the chatbot was particularly helpful or fell short of your expectations?

Form Submission No.	Answer
1	The questions I think would not be necessary to ask course responsible instead I can ask the bot, like when is the deadline of the project submission? How many lectures we have in MG2040? etc.
5	Detailed explanations with examples.
8	Questions in the middle of the night.
7	Mostly unrecognized questions.
17	Easy understanding questions.
20	Repetition.

Table B.2: Answers to the question: Which type of questions would you ask the chatbot as opposed to the teacher or teaching assistants?

Form Submission No.	Answer
1	Not that much because it's still new for us.
3	Made it easier to get topics and what prompt to use for searching a certain type of information needed.
5	Yes, it explains better.
8	I use the chatbot instead of searching through all documents.
7	No, mostly if I have doubts I can access the chatbot.
19	Yes, it's faster.
20	Not really as I forgot to use it a lot.

Table B.3: Answers to the question: Has using the chatbot changed the way you access information for your courses? If so, how?

Form Submission No.	Answer
1	I have had pleasant experience. I thought it would be stupid but it was not!
3	It was helpful.
4	Its fun.
5	Learning is easy.
7	Its nice.
8	Very effective.
20	Good.

Table B.4: Answers to the question: What has been your overall experience using the bot for course-related queries?

Form Submission No.	Answer
1	Yes
2	Yes
3	Yes
4	Yes
5	Yes
6	Yes
7	Yes
8	Yes
9	No
10	No
11	No
12	No
13	No
14	No
15	No
16	No
17	Yes
18	No
19	Yes
20	Yes

Table B.5: Answers to the question: Have you tried the chatbot?

Form Submission No.	Answer
9	It is monitored by the teacher.
10	It was not just a priority of mine till date, but will try using it while navigating in the examination and see how it goes.
12	No need for now.
13	Because I don't take Time to try maybe it's better if you show an example of using during the course.
14	I don't take the time.
16	Had no reason so far might use it for exam preparation.

Table B.6: Answers to the question: Why have you not used the chatbot?

Form Submission No.	Answer
1	Strongly agree
2	Agree
3	Agree
4	Neither agree nor disagree
5	Agree
6	Agree
7	Strongly agree
8	Agree
17	Neither agree nor disagree
19	Agree
20	Strongly agree

Table B.7: Answers to the question: Overall, the information the bot provided to me has been useful

Form Submission No.	Answer
1	Very effectively
2	Very effectively
3	Very effectively
4	Moderately effectively
5	Moderately effectively
6	Very effectively
7	Moderately effectively
8	Very effectively
17	Moderately effectively
19	Moderately effectively
20	Very effectively

Table B.8: Answers to the question: Overall, how effectively has the bot been able to answer your questions?

Form Submission No.	Answer
1	Strongly agree
2	Agree
3	Agree
4	Neither agree nor disagree
5	Agree
6	Neither agree nor disagree
7	Agree
8	Agree
17	Neither agree nor disagree
19	Agree
20	Agree

Table B.9: Answers to the question: Overall, the answers from the bot have been correct

Form Submission No.	Answer
1	Neither agree nor disagree
2	Agree
3	Agree
4	Neither agree nor disagree
5	Agree
6	Neither agree nor disagree
7	Neither agree nor disagree
8	Neither agree nor disagree
17	Neither agree nor disagree
19	Disagree
20	Neither agree nor disagree

Table B.10: Answers to the question: Overall, the answers from the bot contained all the information I needed

Form Submission No.	Answer
1	Much easier
2	Much easier
3	Easier
4	Easier
5	Easier
6	Neither easier nor more difficult
7	Neither easier nor more difficult
8	Much easier
17	Easier
19	Easier
20	Easier

Table B.11: Answers to the question: How would you compare the ease of use of the bot with retrieving information from the canvas room yourself?

Form Submission No.	Answer
1	Somewhat faster
2	About the same
3	Much faster
4	Somewhat faster
5	About the same
6	Much faster
7	Somewhat faster
8	Much faster
17	Much faster
19	Much faster
20	Somewhat faster

Table B.12: Answers to the question: How would you compare the time it takes to ask the bot about the canvas room with retrieving information from canvas yourself?

Appendix C

All answers to the form submitted in LD1000

This includes all answers to the questions in the form submitted to the students in the course *LD1000 Lär dig lära online 2,0 hp*.

ID	Answer
1	Den svarar bra på frågor som handlar om kursen och den hjälper en att hitta i canvas om det är nån specifik sak man söker. Det gör den bra tycker jag.
2	Upplevelsen var bra, den svarade korrekt på alla kurs relaterade frågor som jag ställde.
3	Min upplevelse av att använda chattboten för kursrelaterade frågor var mycket positiv och över min förväntan. Den var skalad och konkret. Gissar att just dens informationsutbud här matchade men utmärkte sig i saklighet och korrekt svar när man kontrollerade mot fakta.
4	Den svarade lite långsamt, men verkade ge korrekt svar.
5	Bra fast alltför grunt.
6	Fick känslan av att kapaciteten finns att besvara frågor om kursen på ett relevant sätt, men något ytligt.
7	Min upplevelse var positiv, det känns kul när man skriver en fråga och man får svar
8	Jag tycker att chattbotten kan svara på frågor om hur många quiz man måste göra osv.
9	Den svarar informativt och detaljerat och avslutar med information om var jag kan läsa mer. Användbart eftersom man får svar så snabbt.

Table C.1: Answers to the question: Vad var din upplevelse av att använda chattboten för kursrelaterade frågor?

ID	Answer
1	Jag testade att fråga hur många quiz som ingick i kursen och den beskrev utförligt hur många den bestod av, hur många rätt man måste få för att bli godkänd samt hur många försök man har på sig. Jag tyckte den var extra hjälpsam genom att den gav en länk till "Examinerande moment" så att man själv kan läsa på mer om de olika momenten i kursen!
2	Jag prövade att ställa "större" frågor som hur bestämt bestämma struktureringen av inlärningen av kursen. Där jag tyckte att svaret var väldigt utförligt och kan klassas som extra hjälpsamt. Svaret var en 6 stegs plan om hur hanteringen och bearbetningen av alla moment och moduler skulle gå till.
3	Chatboten var särskilt hjälpsam och den levde upp till dina förväntningar mer än väntat då den var väldigt konkret i sitt svar och snabb på att hitta det exakta svaret jag sökte.
4	Det som kändes extra hjälpsamt var att man fick en länk till mer detaljerad information (gissningsvis källan den inhämtade information från).
5	Nej.
6	Testade att ställa ett par frågor rent allmänt om examinerande moment och upplevde svaren var tillfredsställande.
7	Jag ställde en bredd fråga bland annat och då svarade den med ett par påstående som jag fick välja bland för att få bättre anpassat svar vilket jag uppfattade som hjälpsamt.
8	Den svarade på vad man behövde göra för att bli godkänd i kursen.
9	Den är bra på att ta mig vidare när jag kört fast. Då kan jag fråga boten vad jag ska skriva nu, vilket den förstår inte svarar på men den förtydligar ämnet och säger "tänk till exempel såhär", och så är jag igång igen.

Table C.2: Answers to the question: Uppstod det situationer där chatboten var särskilt hjälpsam eller där den inte levde upp till dina förväntningar?

ID	Answer
1	Det finns egentligen inte någon speciell fråga jag hellre ställer till boten än en lärare. För mig handlar det kanske mer om att jag hellre skriver till boten för att få ett svar på direkten och slipper vänta!
2	Till viss del, särskilt om det är relativt simpla frågor som jag bara vill ha ett snabbt svar på.
3	Jag uppskattar generellt mer att ställa frågor till lärare men tycker att en chattbot är som fungerar som denna skulle vara användbar för att studera på tider när läraren ej är aktiv. Då gällande alla frågor.
4	Man vill ju inte störa kursassistenter eller lärare i onödan för att fråga frågor om kursupplägg, deadlines, etc., så då kändes chattboten som rätt ställe att söka på först.
5	Frågan, varför då?
6	Nej! I alla situationer föredrar jag levande människor. Många onlinekurser har så pass få fysiska onlineträffar, så dessa är oerhört värdefulla. Intressant med mänskliga möten i sig, att få bekantskap med ansiktena bakom en kurs, och att vid behov kunna ställa frågor direkt till dem.
7	Nej, det tror jag inte, frågor kan jag ställa till både lärare, lärarassistent och chatboten? Det som är viktigt är att man kan lita på att det blir rätt svar och att inte chatboten svarar på vad som gäller för andra kurser tex om man frågar vilka examinerande moment som finns.
8	Nej, det tror jag inte. Kanske så kallade "dumma" frågor som borde vara självklara kanske man hellre frågar chatbotten.
9	Är den här diskussionen obligatorisk?

Table C.3: Answers to the question: Finns det frågor du hellre ställer till chatboten än till lärare eller lärarassistent? Vilka?

ID	Answer
1	Det tror jag. Jag hade använt mig av den för att snabbt få fram den informationen jag vill ha. Vilket kan vara störande i kursen som är stora och har många moduler att navigera igenom. Då hade jag definitivt använt mig av boten.
2	Den kan nog hjälpa till med effektiviteten en del med ställandet av simpla frågor och kolla upp vad som examineras på och så vidare.
3	Om jag visste att det var tillåtet skulle jag använda chattboten 100% till att ta in information då den verkar kunna ta ut det viktiga och formulera text som ger bilden av vad materialet gör.
4	För min del föredrar jag nog att ha all relevant information koncist beskriven och samlad på en sida på Canvas, men om informationen är väldigt spridd kan jag tänka mig att chattboten skulle kunna vara till hjälp.
5	Ja. Om den kan bli en gruppkompis som aldrig är sur och har oändligt tålamod.
6	En genomarbetad studiehandledning, vilket ingår i de flesta onlinekurser är fullt tillräckligt, då behövs ingen chatbot. Tilltalas personligen av att ha all information samlad om en kurs i en studiehandledning.
7	Nej det tror jag inte, men det är egentligen svårt att svara få jag använt det för lite för att ha någon egentlig uppfattning.
8	Nej, inte väsentligt. Men det kan göra att man snabbare får svar på sina frågor än om man frågar en lärare.
9	Om det är något jag inte förstår kan jag fråga och få ett sammanfattat svar. Då kan jag välja om jag vill läsa mer utförligt eller inte beroende på ämnets relevans.

Table C.4: Answers to the question: Tror du användningen av Chatboten kan ändra hur du tar till dig information i en kurs? Hur då?

ID	Answer
1	Jag tycker att AI-verktyg är en bra grej och använder det i mina studier. Främst använder jag det för att få saker förklarade för mig på andra sätt som kanske är lättare för mig att förstå. Vissa lärare kanske undervisar på ett sätt som inte passar mig. Då är det smidigt att kunna lägga in deras föreläsningar i ett sådant verktyg och be verktyget att formulera om det eller förtydliga otydliga delar.
2	Jag kan se det positiva i att utnyttja AI-verktyg i mina studier men använder det för tillfället inte i mina studier idag.
3	Jag använder endast AI ifall jag vill hitta något jag inte direkt gör i min litteratur men använder den mera privat för att ställa frågor om diverse saker.
4	Jag använder inga AI-verktyg i mina studier och känner nog inte riktigt att det är där de gör mest nytta heller (och då arbetar jag ändå med AI, och har en generellt positiv inställning till det).
5	Chat gtp för programmering.
6	Jag använder inga AI-verktyg i mina studier. Tycker om att vara kreativ själv.
7	Jag använder inte några verktyg idag men jag tycker att det är intressant och testar gärna olika.
8	Jag tycker det är bra, jag använder det ganska ofta för att få en förklaring på något som jag tyckte behövdes förtydligas. Jag använder det också när jag skriver rapporter som ett avancerat rättstavningsprogram som kan hjälpa så att det blir en bra meningsuppbyggnad och låter bra.
9	Jag har inte haft för vana att använda AI-verktyg. Jag vet att de används för att formulera text ibland, vilket jag tycker är lite fusk, men samtidigt en ganska bra idé eftersom så många har svårt för just det.

Table C.5: Answers to the question: Hur tänker du kring användandet av AI-verktyg i dina studier? Använder du några verktyg idag? Vilka, till vad?

Appendix D

All answers to the form submitted in LD1006

This includes all answers to the questions in the form submitted to the students in the course *LD1006 Kognitiv psykologi för lärare: Matematikundervisning 3,0 hp*.

ID	Answer
1	För enklare frågor fungerar botten bra.
2	Jag är helt ny användare så det är svårt att säga något nu :)
3	Använde dem bara för att svara på AI verktyg frågorna.
4	Jag är relativt ny i användandet av AI, därför prövade jag lite olika inlägg i AI-Copilot.
5	Jag frågade om zoom-mötena spelades in och lades upp på sidan - jag fick svaret ja. Så där ger den mig fel information.
6	Det fungerade bra tycker jag. Ställde bara enkla frågor som jag redan hade svaret på.
7	Den svarar väldigt svepande och allmänt.
8	Frågade och fick ett korrekt svar.
9	En relativt positiv upplevelse.
10	Den var oväntat positiv! För mig var det inte självklart med kopplingen mellan hämtningskapacitet och lagringskapacitet vilket jag ställde en fråga om. Jag fick ett utmärkt svar och förstod bättre än innan.
11	Aldrig använt AI chat tidigare, men tycker det fungerade riktigt bra. Jag frågade bland annat om hur man undervisar lågpresterande elever och fick då både sammanhängande text och även en lista med råd på hur man kan arbeta med dessa elever. Jag frågade också om "den tysta läraren" och fick då svar på min fråga.
12	Bra. Jag fick svar på min fråga.
13	Bra upplevelse. Den hjälpte mig att besvara på frågor kring sena inlämningar.

Table D.1: Answers to the question: Vad var din upplevelse av att använda chattboten för kursrelaterade frågor?

ID	Answer
1	För frågor som är relaterade till självaste kursmaterialet, dvs teoretiska frågor, är den näst intill inkapabel.
2	Det är bra att få bra verktyg, men det är bra tiden som visar om det är bra/effektivt eller inte.
3	Ser ut som det kan bli hjälpsamt när det gäller definitioner. Men för djupare frågor var kritisk tänkande krävs, chatbot svarar igen baserat på kursens definitioner.
4	Jag skrev några olika inlägg med fråga om lärstilar kopplat till forskning och önskade svar skrivet av en 15-åring och senare ett svar skrivet av en forskare. Det var intressant att se hur boten anpassade texterna utifrån min önskan. Texten var välformulerade och anpassade utifrån vem som enligt min önskan skulle vara författaren. Dock fick jag endast en källhänvisning till forskning.
5	Jag frågade om zoom-mötena spelades in och lades upp på sidan - jag fick svaret ja. Så där ger den mig fel information.
6	Jag kunde söka efter ett specifikt avsnitt i boken för att få veta vilket kapitel jag skulle leta i. Det hade jag inte förväntat mig.
7	Jag har inga direkta förväntningar.
8	Kan inte säga att jag tog det vidare. Fick inte svar på djupare frågor i kursen.
9	Det enda var väl att den någon gång svarade på engelska fast frågan ställdes på svenska. Sen svarade den antingen ganska kort och inte så uttömmande som andra chatbotar på vissa frågor och andra frågor svarade den inte på hela frågan. Exempelvis ställde jag fråga om den kunde nämna 3 inre samt 3 yttre motivationer, då nämnde den enbart 3 inre.
10	Nej det var jag inte, den var precis så hjälpsam som jag behövde.
11	Chatboten klarade att svara på alla de frågor jag ställde på ett tillfredsställande sätt.
12	Vet ej, har bara testat 1 gång.
13	den var mycket hjälpsam.

Table D.2: Answers to the question: Uppstod det situationer där chatboten var särskilt hjälpsam eller där den inte levde upp till dina förväntningar?

ID	Answer
1	Teoretiska frågor från kursmaterialet.
2	Svårt att svara på den frågan nu.
3	Nej, hittills inte alls.
4	Svårt att säga eftersom chattboten är helt ny för mig.
5	Nej - är väl allmänt lite skeptisk till olika chatbotar (även på andra diverse sidor där man kan chatta med en chatbot). Jag mailar eller ringer hellre till en person för att få svar.
6	Detta kan jag inte svara på i nuläget.
7	Nej.
8	Nej.
9	Inga specifika frågor men jag kanske skulle välja att ställa frågor till chatboten i första hand då jag tänker att jag får svar fortare av den då den inte blir lika överbelastad som lärare kan vara.
10	Kanske om det var frågor jag kände mig dum att fråga.
11	Åh, svårt, men det finns det säkert. Frågor som man kanske tycker att man ska ha koll på ställer man säkert hellre till en chatbot än till en lärare.
12	Frågor om kursen, administrativa frågor är bra att ställa till chatboten. Man får ju svar direkt vilket underlättar mycket.
13	var inte helt säker på till vem man skulle vända sig till för frågor så då var det bra att den här fanns som kunde svara istället.

Table D.3: Answers to the question: Finns det frågor du hellre ställer till chatboten än till lärare eller lärarassistent? Vilka?

ID	Answer
1	Nej.
2	Det är svårt att säga eftersom chattboten är helt ny för mig.
3	Jo, kanske om, i framtiden, jag kommer inte att hinna läsa hela kursens material inom kort tid.
4	Det är svårt att säga eftersom chattboten är helt ny för mig.
5	Nej - inte som den fungerar nu.
6	Kanske framöver när jag lärt mig mer om det. Just nu litar jag inte fullt ut på att det fungerar.
7	Nej.
8	Nej egentligen inte. Jag har personligen ringa intresse för detta förutom hur mina elever kan tänkas använda det. Så i min profession är jag intresserad men för mitt eget lärande - nej.
9	Beror på vilken slags kurs det är och hur gammal chatboten är. Den kan vara användbar om man ställer frågor kring ex marknadsföring men jag ställde en enkel mattefråga till den nyss och den svarade tyvärr fel, liksom ChatGPT 3,5 gjorde.
10	Det vet jag inte i dagsläget.
11	Ja, det kan ju eventuellt förtydliga begrepp som man tycker är svåra att förstå, samt evt underlätta informationssökning.
12	Vet inte, har inte så mycket erfarenhet.
13	Den kan förtydliga saker och ting för mig vid arbetet med case och quiz.

Table D.4: Answers to the question: Tror du användningen av Chatboten kan ändra hur du tar till dig information i en kurs? Hur då?

ID	Answer
1	Jag ser inte speciellt positivt på AI-verktygen just när det kommer till matematik, för i slutändan behöver man "plöja" problem som man själv behöver förstå, applicera teori och tänka kring. Men det är möjligt att det är användbart i andra ämnen och områden. Därför ser jag personligen ingen användning av det idag.
2	Jag gick på en kurs för en månad sedan där jag fick veta om AI-verktyg, det låter spännande och skrämmande samtidigt :)
3	Jag är inte ett stort fan av chatbots. Jag märkte en gång att några av mina elever använde det för att lösa ordproblem; det gjorde inte ett bra jobb så det fortsatte inte med det.
4	Min fundering är hur man som lärare ska kunna se eller kontroller om en text är författad av en student eller av en AI? Jag testade att skriva in samma fråga två gånger och fick olika svar. Som jag ser det är det inte en fråga om användandet av AI eller inte. Det kommer att användas även i studiesammanhang vare sig vi vill eller inte. Jag tänker att via AI-verktyget kan vara en hjälp för att få en snabb koll kring en fråga. Den kan också hjälpa till att förstärka det man läser och lär sig, men den kan inte, i alla fall för mig, ersätta den lärprocess som sker när man tar till sig kunskap jag läser (helst analog) i kurslitteratur eller studier samt det som sker i kunskapsutbyte med andra personer. Jag är ännu för okunnig om AI-användning för att tänka kring hur det kan användas i undervisning inom matematik.
5	Jag undervisar i en kurs på gymnasiet för våra åk 3 elever som heter Ledarskap och organisation. Där är AI inte med i det centrala innehållet, men något som jag tycker bör lyftas i och med att det tar större och större del i vår vardag men även arbetsliv. De får diskutera och intervjua (i samband man annat område med) hur ett företag/organisation använder sig av AI - men även nu på slutseminariet får de diskutera utifrån vad de själva tycker och om de t.ex. vill byta ut chefen mot en AI (utifrån en artikel de läst). Även viktigt att förklara för dem hur t.ex. chat GPT fungerar (för alla har använt den till skolarbete på olika sätt), att lära dem hur de kan använda den på ett smart sätt som en "studybudy", men att vara lite mer vaksam när det gäller information m.m. Men en jättebra källa om man söker inspiration till något, skapa quizar m.m.
6	Jag provade att tillverka läsförståelsefrågor åt eleverna med hjälp av ChatGPT. Det såg väldigt bra ut till en början men när man läste mer noggrant fanns det ingen som helst substans i texten. Förmodligen för att jag ställt frågan på helt fel sätt. Jag är inte negativ till AI-verktyg men känner att jag måste lära mig mer innan jag kan/vågar använda det i undervisningen.
7	Ja, jag har använt AI för att skapa flera varianter av enklare matteprov. Men man måste alltid modifiera. Jag ställer ibland frågor. Eleverna använder det för att svara på frågor ibland, men man kan se att det inte är deras egna ord ganska ofta. Det kommer nog att bli stort i framtiden. Men just nu är vi inte riktigt där, är känslan.
8	Jag vet att många elever, särskilt på gymnasiet, använder det. De använder det mest för att få jobbet gjort och lämnar in. Här är det viktigt att vi lärare lär oss om hur det fungerar. Här blir det liksom en kollaps i lärandet som jag ser det. Vi pratas inte ens om prestation kontra lärande utan en AI som levererar åt dig. Problematiskt skulle jag säga. I matematiken anser jag att det är ett mindre problem men i andra ämnen med tex skriftliga inlämningar så behöver man tänka över sin bedömning som lärare. I matematiken skulle det kunna vara ett stöd för elever i det att de får exempel på lösningar till matematiska problem som de sedan kan överföra till andra uppgifter. Eleverna behöver dock få tillgång till studiestrategier som ger dem faktisk nytta av den information AI kan ge dem.
9	Med mina elever använder jag inte AI-verktyg och då jag mest håller på med matte så tror jag inte att jag kommer göra det heller inom en snar framtid. Däremot brukar jag själv använda mig av ChatGPT 3,5 (tycker USD $20/mån är lite väl dyrt för 4,0, men ska nog inom en snar framtid börja använda den istället). Håller på att starta upp ett företag och då tycker jag ChatGPT är användbar vid frågor kring hur man kan attrahera kunder, marknadsföring, mm.
10	Idag använder inte mina elever några sådana verktyg och det beror nog på att jag är ovan vid och okunnig om det själv. Men jag ska lära mig mer! Först då kan jag svara bättre på den frågan.
11	Jag undervisar just nu i en årskurs 1 så AI används inte av mina elever. Inte heller av mig i min yrkesroll som lärare. Jag skulle dock absolut kunna tänka mig att använda mig av AI-verktyg om jag fick utbildning i hur det ska användas och vilka fallgropar man bör undvika.
12	Jag använder i nuläge väldigt sällan AI-verktyg, men jag har testat att göra en planering till en schemabrytande dag med fokus på motivation. Det gick ganska bra. Jag fick bra tips på aktiviteter, och det gick snabbare än att googla själv. Eleverna får inte använda AI-verktyg just nu, men vi har haft diskussion om det. Jag är öppen för att testa nya AI-verktyg.
13	Jag använder det ibland till hjälp med att formulera uppgifter. Jag vet att också elever använder det bland annat för att söka svar på uppgifter som de sedan kan öva på.
14	Jag missade att använda denna chattbott men brukar uppskatta när det finns. Finns det frågor du hellre ställer till chatboten än till lärare eller lärarassistent? Vilka? Är det frågor men relativ enkla svar så skulle jag gärna ställa dem till en chatbott. Känslan av att fråga något som redan sagts av lärare minskar om man har en chatt att fråga. Jag brukar uppskatta att ta reda på saker själv och genom en chattbott så kan frågorna /svaren utveckla sig jämfört med bara färdiga fråga och svar. I detta läget tror jag inte att man kommer att ta åt sig information på annat sätt. Vi använder i nuläget inget AI-verktyg. Bland mina elever så tror jag att det är viktigt att de förstår hur AI -verktyg fungerar och har ett källkritiskt tänkande. Vidare så tror jag att arbetsmarknaden kommer att ändras och att det tex blir viktigt att kunna /förstå att skriva tex promptar.

Table D.5: Answers to the question: Hur tänker du kring användandet av AI-verktyg både av dig själv och dina elever? Använder du och/eller dina elever några verktyg idag? Vilka, till vad?

TRITA-EECS-EX- 2024:0000