EduChat: Building a Chatbot for Educational Purposes

Project in Conversational Systems
DT2151

Simone Clemente - simonecl@kth.se
Elin Saks - esaks@kth.se
Tora Wallerö - toraw@kth.se

KTH Royal Institute of Technology
School of Electrical Engineering and Computer Science

January 2024

Abstract

This study explores the implementation and evaluation of EduChat, a chatbot built for educational purposes. Drawing on the challenges posed by the diverse needs of students and schools' reluctance towards AI, the research leverages Large Language Models (LLMs) to create a student-centered learning tool that is safe and easy to use. This chatbot exclusively references and summarizes information from a designated document to mitigate concerns about AI-generated hallucinations, thus functioning as an extension of a textbook. The goal is to offer correct and concise answers with conversational characteristics, promoting effective learning experiences.

The chatbot operates in two modes: as a tutor providing explanations and engaging in discussions, and as an examiner, quizzing students on a subject. The methodology involves prompt engineering to tailor the language model's responses and building a chain for retrieval-based enriched conversations. The evaluation, consisting of automatic analysis and user testing, found the product user-friendly while concisely providing relevant questions and responses. These encouraging results open the possibility for actual implementation of this technology in the education field.

1 Introduction

Self-driven learning and self-assessment are two important components when building knowledge. As shown in a study by Robinson and Persky (2020), such self-centered learning, achieved through reflection and the pursuit of new perspectives and insights, is employed in today's educational systems. However, implementing such learning approaches in schools can be challenging due to a lack of staff and the diverse needs and desires of students. One potential solution to address this challenge is leveraging the rapid development of generative AI. A concern with generative AI is the potential for generated responses to appear confident while containing nonsensical or wrongful statements, a phenomenon known as hallucination, described by Alkaissi and McFarlane (2023). These limitations, combined with the fact that the school staff has a general reluctance regarding such technologies, are slowing down the mass adoption of these models by educational institutions even though, chatbots such as ChatGPT are widely used among students during their studies.

This project aims to develop an educational chatbot aiding students in their learning process. The tool should give correct and concise answers while also having conversational characteristics. To ensure that the chatbot gives correct answers that are free from hallucinations it should exclusively reference and summarize information from a given document, functioning as an extension of a textbook. The goal was to create a tool featuring two modes: one where the chatbot serves as an examiner, quizzing the student on a subject, and another where it acts as a tutor, offering explanations in response to a question. The tutor should also be able to engage in a discussion on a topic, rather than just generating an answer.

To create this student-centered learning tool, prompt engineering on a large language model, GPT-3.5 from OpenAI, was used to construct a teacher-like chatbot, capable of supporting students. The LMM was prompted to solely use information derived from a given textbook to ensure that the responses contained accurate information.

2 Related works

2.1 Educational Chat-bots

Kuhail et al. (2023) have reviewed a lot of chatbots for educational purposes and analyzed how they are designed and used, and their effect on learning. They found that the majority of chatbots developed in recent years are teaching agents applied to various educational areas, most commonly computer science-related topics. Some were peer agents, meaning they could answer some questions from the students instead of only presenting topics for discussion or giving multiple-choice questions to evaluate the student's abilities. The results show a lack of chatbots asking students open-ended questions. Further, most of the chatbots were chatbot-driven, which means that the chatbot was the one leading the conversation and doing so by following a predetermined path. Others were intent-driven but also relied on a few predetermined answers the chatbot could choose from.

With the rise of LMMs, there have been more chatbots created that do not follow a strict script. One example is the teacher chatbot by Ali et al. (2023), which uses a LLM by OpenAI that they prompt engineered to support self-directed learning. Kuhail et al. (2023) found evidence for improved learning, through experiments, in some of the studies they analyzed. Okonkwo and Ade-Ibijola (2021) also reviewed chatbots for educational purposes and found several articles supporting the conclusion that such chatbots improve learning outcomes as well as keep the students engaged. Another advantage was the accessibility and always being there to support students, unlike human teachers who have many other students to tend to. They also identify potential issues when using these types of chatbots, like making sure their information is correct and up-to-date and ethical issues such as transparency, trust and privacy.

2.2 GPT-architecture

Generative Pre-trained Transformer (GPT) is a type of transformer that is auto-regressive and decoder-only, meaning that the model automatically predicts the next output by relying on the previous input where the input is fed through a decoder, without being encoded first. Being a transformer, the model's main characteristic is the self-attention mechanism which decides which of the previous words in the input to focus on when creating the output. One contributor to the latest year's development of GPT models is OpenAI with their latest release of GPT-4 in 2023, presented in a report by Achiam et al. (2023).

2.3 Prompting

Prompt engineering is a technique used to tailor a LLMs output by instructions. Prior research has studied different methods on how to create effective prompts, guiding the LLM into desired behavior. Brown et al. (2020) demonstrates that prompts can be used, both with examples (few-shot) and without (zero-shot) to improve the performance of OpenAI's GPT model given a specific task. They conclude that prompts can be used rather than fine-tuning to get the desired behavior, even though the language model is not specifically trained on that task.

Another important field of study is how to construct prompts methodically. White et al. (2023) present guidance on how to construct a successful prompt by providing a prompt pattern. The article also provides a catalog of prompts tested and evaluated on a range of tasks, giving insight into how to create a strong prompt.

3 Method

3.1 PDF Pre-processing

Working with a retrieval-based chatbot implies the creation of a collection of documents from different possible sources. In our case, we opted for school books which can be found in PDF format. The problem is that these PDF files do not contain only text: they can contain images and text which is not searchable so it cannot be read by a classic loader. For this reason, it is always advisable to pass the PDF file through an OCR (Optical Character Recognition) process which can create a fully searchable file.

3.2 Documents Indexing

Once we have a searchable PDF we can extract the actual text from it: the goal is to create an indexed collection of text documents which allows us to retrieve useful information for the context needed to handle a specific topic. To address this task we relied on the FAISS library. Given a set of vectors x of dimension d, FAISS generates a data structure in RAM. To do so we need to rely on an embedding function. After the structure is constructed, when given a new vector we are able to perform an efficient similarity search.

The reason for the indexing process is being able to retrieve the information related to the input: every time we receive a message from the user it is fed to the embedding function which generates the related index. For this reason, it is crucial to use the same indexing function for both documents and inputs: in this way, we can generate coherent embeddings which can lead to a meaningful search process.

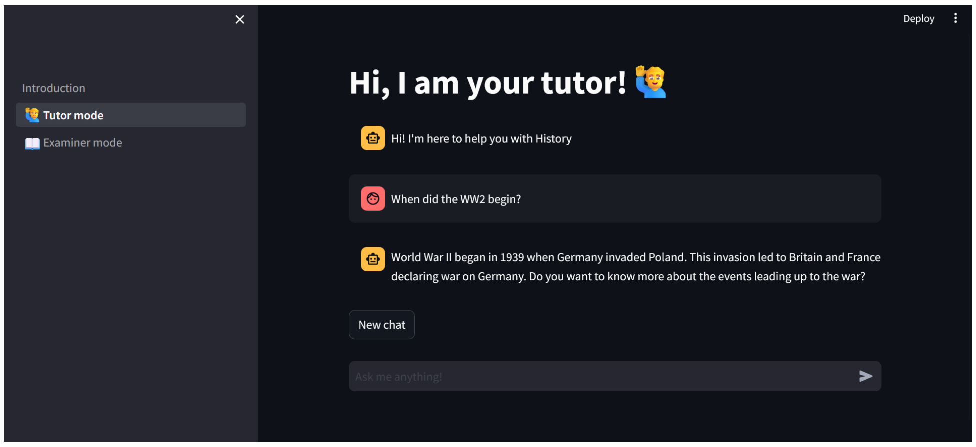
Similarity search is performed by default using L2 norm, which returns the k most similar elements according to the formula:

$$elem_j = argmin_i ||query - elem_i||$$

Alternatively, it is also possible to exploit maximal marginal relevance (MMR) which is a method used to diversify search results by selecting documents that are both relevant to a query and dissimilar to the documents already selected. To do so a penalty term dependent on the similarity is applied to already selected chunks:

$$MMR(x) = \lambda sim(x, y) - (1 - \lambda) max_{i \in elems} sim(x, x_i)$$

The λ term controls the trade-off between relevance and diversity: a high λ implies a behaviour close to the standard. In the final implementation, we decided against using MMR.



Hi, I am your tutor!

Hi! I'm here to help you with History

When did the WW2 begin?

World War II began in 1939 when Germany invaded Poland. This invasion led to Britain and France declaring war on Germany. Do you want to know more about the events leading up to the war?

Figure 1: Tutor interface

3.3 Tutor

The first section of our interface is composed of a tutor chatbot which is able to help students by taking information directly from the assigned textbook. The student can ask questions about a topic and the chatbot answers and try to keep a conversation with the user.

3.3.1 Text Generation

For this task, we had to generate a conversational chatbot able both to retrieve information and to keep the conversation context. In a simple QA chatbot, each question is tokenized in order to retrieve a matching context from documents. However, in a conversation environment, using the simple user question to perform the retrieval task might not be enough: conversation poses challenges such as anaphoras and ellipsis which must be handled properly before performing the similarity search. If we do not address this issue we will end up with inconsistent documents for the context. Let us imagine we are discussing Word War II and at some point, we ask “When did it begin”. The utterance *it* is not meaningful by itself since the model does not know the actual meaning of “it” if extrapolated from the context. A possible solution is to generate a chain and divide the LLM action into two steps: question reformulation and retrieval-based answer.

As a first step, we perform a first call to the LLM exploiting the user question and the chat history: this part aims to reformulate the question using the history as context to make it standalone. We will get as a response from the LLM a new history-aware standalone question which can make sense on its own. Once we obtain a meaningful user utterance, it can be used for similarity search in our indexed documents.

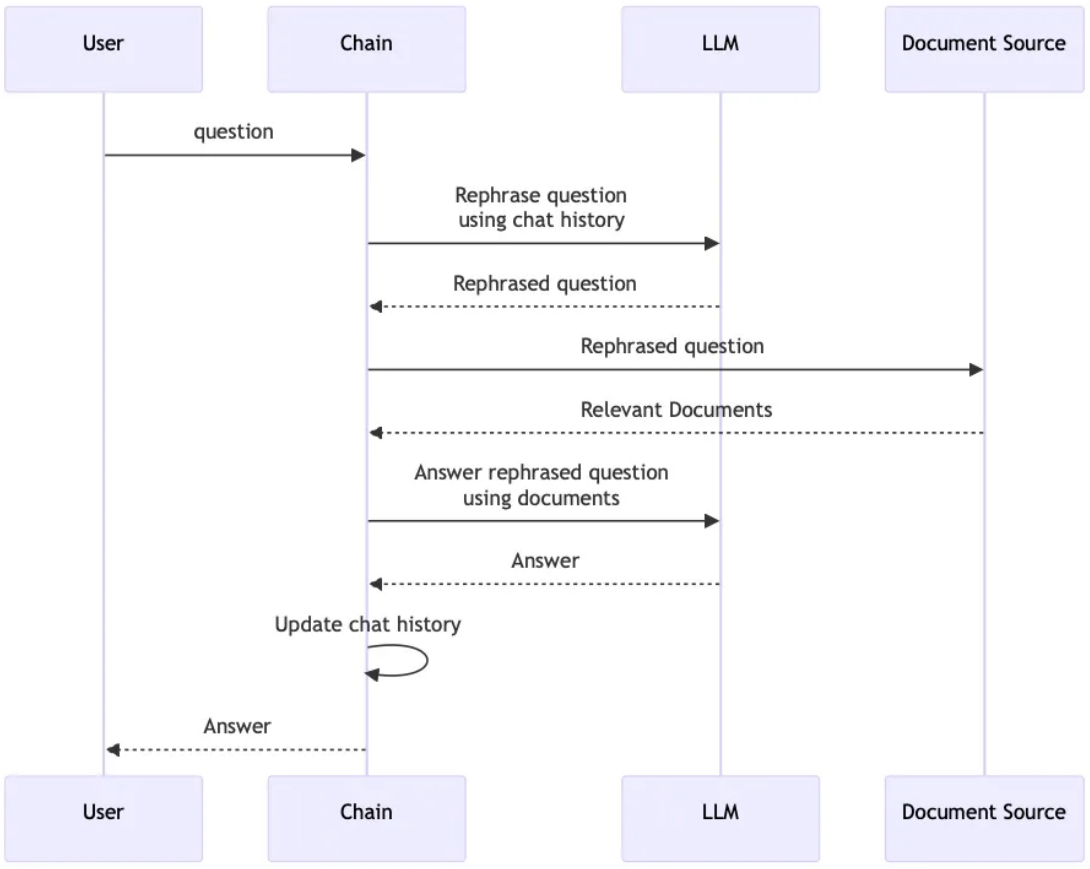


Figure 2: Tutor model logic

3.3.2 Prompt Building

For the tutor chatbot, the following zero-shot prompt is used to generate a response to the user:

```
You are a history tutor having a conversation with a human.
You knowledge is extracted from a document on world history between
1910 to 1945.
Given your knowledge and an input from the user, have a conversation
```

```
on the topic.
When asked a question,create a concise answer and a question to
continue the conversation. The answer offer explanations, and provide
summaries from the context. If the user input says which it do not
want to continue the conversation stop it saying you are glad to
help. Make sure that the user understands the given answer, by asking
a follow up question. If the user answers to the follow up question
saying he does not want any other information say that it is okay and
you are there for any further help. You cannot have political
influence and you should be neutral when asked about subjective
opinions. When you cannot find information in the context answer that
you don't know. You are forbidden to answer questions on topics not
included in the context.

Context: {context}

Question:{question}

Answer:
```

The prompt is customized to give the LLM characteristics such as giving concise a answer, making sure that the user understood the answer and to only include information provided in the given document. The creators of the used LLM, OpenAI, provide a guide, OpenAI (2023), on how to improve the desired performance of the LLM by prompt engineering. They suggest asking the model to adopt a persona, this is adopted in our prompt by telling it to behave as a tutor. The prompt used in the project also includes restrictions such as that only information retrieved from a given document should be used when constructing an answer and that the chatbot should be objective. The conversational properties are also described in the prompt to ensure that the chatbot asks follow-up questions and engages in the discussion.

A second prompt is used to rephrase the input question before retrieving relevant parts of the document:

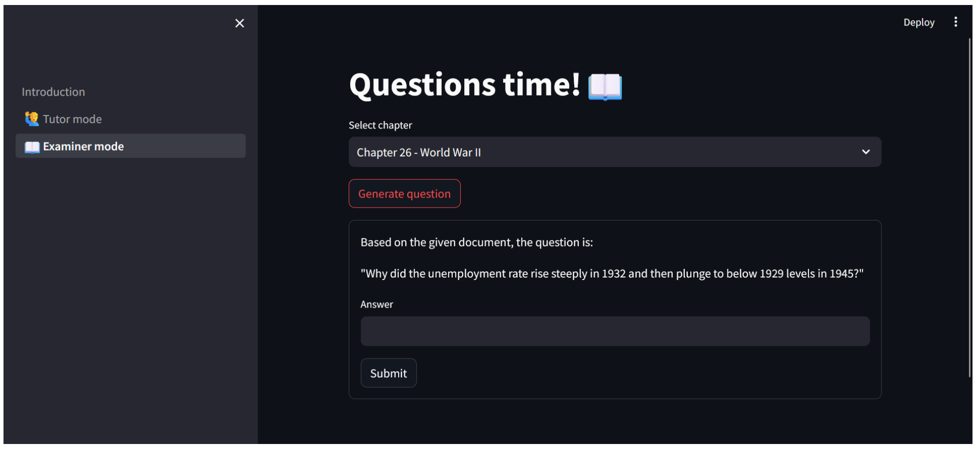
```
Given the following conversation and a follow up user input,
rephrase the follow up user input to be a standalone sentence. If in
the last chatbot answer the user was offered more information and in
thefollow up input he refused say "No, I want to end the
conversation".

Chat History: {chat_hisotry}

Follow Up Input: {follow_up_question}

Standalone sentence:
```

Here the chat history is utilized to make sure that the follow-up question contains all information needed to be understandable on its own. A sentence to instruct the LLM to end the conversation when the user is done was also added to improve the performance.



Questions time! 📖

Chapter 26 - World War II

Based on the given document, the question is:

"Why did the unemployment rate rise steeply in 1932 and then plunge to below 1929 levels in 1945?"

Figure 3: Examiner interface

3.4 Examiner

The second use case is the examiner mode: this function allows the user to practice and test their knowledge through automatically generated questions from a chapter of choice. Once the student answers the generated question, the model evaluates it while giving an explanation. For this section we generated a list of indexed collections, each one referring to a specific chapter, see section 4.3 for more details about the chapters. Two separate models were used for this functionality: one generating the question and one evaluating the answer. Both of them rely on retrieving information from the book but they have different prompts to make them behave as intended.

3.4.1 Question Generation

The rationale underlying the generation of questions involves providing the model with contextual information from a designated chapter. Leveraging a tailored prompt, the model is instructed to generate questions pertinent to the specified topic. Users are tasked with selecting a topic of interest, and for each question request, we extract k continuous random segments from the chapter. This methodology allows us to furnish the model with diverse contextual inputs, thereby enhancing the variability of our generated outputs.

3.4.2 Answer Evaluation

Again using the contextual information from the designated chapter used to create a specific question for the user, the answer to that question, provided by the user, is evaluated by a model. The model is provided with the question, the answer and the context through which it decides if the answer is correct or not. A response text is generated through a specific prompt that tells the model to give feedback on the provided answer and explain the correct answer.

3.4.3 Prompt Building

To generate a question the following prompt is used:

```
You are a teacher who will enhance my knowledge through quizzing.
You will teach by posing questions on a subject of my choice.
Please create an open-ended question based on the following
```

```
document, do not refer to any images or tables present in the
document.

Please also provide some context from the document before asking the
question so that the user understands where the question is coming
from.
The answer to the question should NOT be given in the context.
The underlying document should be mentioned in neither the question
nor the context.

{context}

Chatbot:
```

The focus is on creating questions but we also wanted the user to get some context since there are lots of information in each chapter. Further, an attempt is made to discourage the response from containing phrases like "In the document..." or "The provided context...".

The evaluation prompt is based on another template:

```
You are a teacher who will enhance the user's knowledge through
quizzing.
You will facilitate their learning by offering hints, clues, and
suggestions for clearer explanations when the user struggles to
answer fully.

The question you gave the user was: {question}
User answer: {human_input}

Please evaluate the answer by comparing it to the information in
the following book: {context}
```

We tried to keep the prompts short and provided the necessary variables. The guides and inspiration for the prompts were the same as for the tutor, see section 3.3.2

3.5 Evaluation

3.5.1 User Testing

User testing was carried out on 20 people in the age range 18-30 to test the chatbot and evaluate its behaviour. Users were told a bit of background information about the product, the underlying history document and the intended usage. They were then free to try it out under the supervision of one of the group members. The testers first spent some time on the tutor and then moved on to the examiner mode. After feeling like they had used it enough, an evaluation form was filled in. The form consists of 10 statements that the tester can agree or disagree with on a 7-point Likert scale, ranging from "Strongly disagree" to "Strongly agree". There were also two free-text questions where the tester could add additional comments and one question about how often they use ChatGPT for studying.

3.5.2 Automatic Evaluation

We wanted to back up our user testing results with some quantitative measures so we did two types of automatic evaluation of the tutor. The first one was comparing the average word count of a response to that of ChatGPT's when asking the same question. We did this to evaluate the conciseness of the responses. The other part was counting the number of responses ending with a question as a form of measure of the level of conversation or mixed initiative.

4 Implementation

4.1 LLMs configuration

The project implements the LLM functionalities through the OpenAI API service. This choice has been made since the platform has different models available and they are perfectly integrated with the `Langchain` library which was used as a base for the implementation of the chatbot. As previously stated we developed our project by using GPT-3.5-turbo since it is the same model which powers the free ChatGPT version making it perfect for our comparisons. The tutor model was initialized with temperature 0.2, in order to add some variance to the outputs but, at the same time, to stick as much as possible to the book. For the examiner, a temperature of 0.5 was used. The higher temperature was needed to generate a variety of questions, however, for the evaluation model, a lower temperature could be an alternative.

4.2 Interface

Users can interact with the model through a user interface built around the `streamlit` library: this package offers the possibility to build markdown-based web applications with modern design and diverse functionalities. The layout is divided into pages which allow the user to change mode. The tutor mode offers a typical chat-based view where the user can have a conversation with the model. The entire history is saved and the user can scroll up to read the conversation. A button is present to delete the current chat: this action will act both in the front-end and the back-end, offering a whole new context for the next interaction. On the contrary, the examiner part is more guided: the user can select the chapter using a drop-down menu and by pressing a button a question is generated. Once the student inserts an answer and presses the submit button, the model outputs an evaluation of the given answer.

The interface has been created to be easy to use and with few distractions to help students concentrate on their studies and maximize their performance.

4.3 Selected Documents

The documents used for the evaluation are chapters 23-26 of the book Glencoe World History by Spielvogel (2010). Each chapter is 30-40 pages resulting in a total of close to 150 pages. The selected chapters cover world history during the period starting from the beginning of World War I to the end of World War II. The book is appropriate for grades 9 and up.

Chatbot	Avg. word count	Follow-up
EduChat	72 words	81.25 %
ChatGPT	261 words	0.00 %

Table 1: Automatic evaluation results

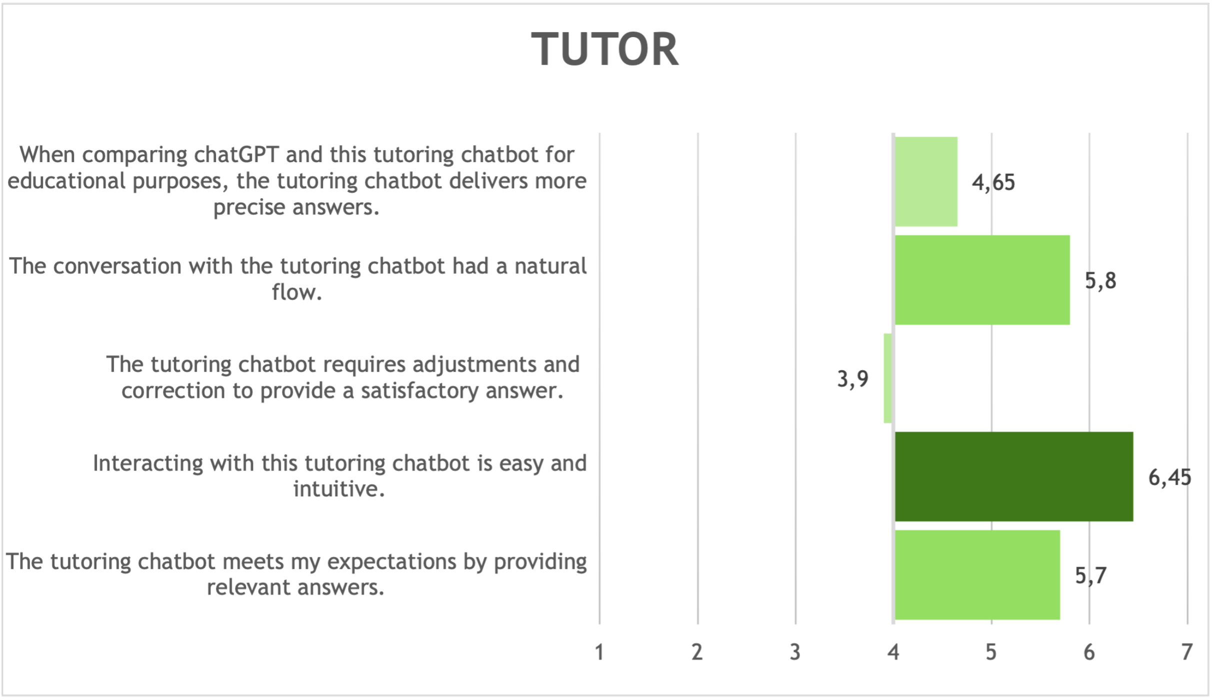


Figure 4: Average scores for the Tutor questions on a scale from *1 - Strongly disagree* to *7 - Strongly agree*

5 Results

5.1 Automatic Evaluation

The results are presented in table 1 and show that the average word count for EduChat is less than that for ChatGPT. On average, EduChat answers have a total length that is less than 30% of the length of answers given by ChatGPT. No follow-up questions are provided by ChatGPT, whereas more than 4 out of 5 times our chatbot ends the response with a follow-up question.

5.2 User Testing

According to our study 75% of the testers use ChatGPT for studying and almost half of them use it daily. The average scores of agreement with the statements about the tutor are shown in figure 4. Please note that statement 3 is a negative statement and thus a low score is desired, hence the bar is green despite the score being below 4. For a majority of questions, the average tester at least somewhat agrees with the statements.

The results from the examiner questions are presented in figure 5. For all statements, the average score is very close to 6 which indicates "agree". The examiner scores particularly high on being easy to use. Most testers think this could be a useful tool when studying.

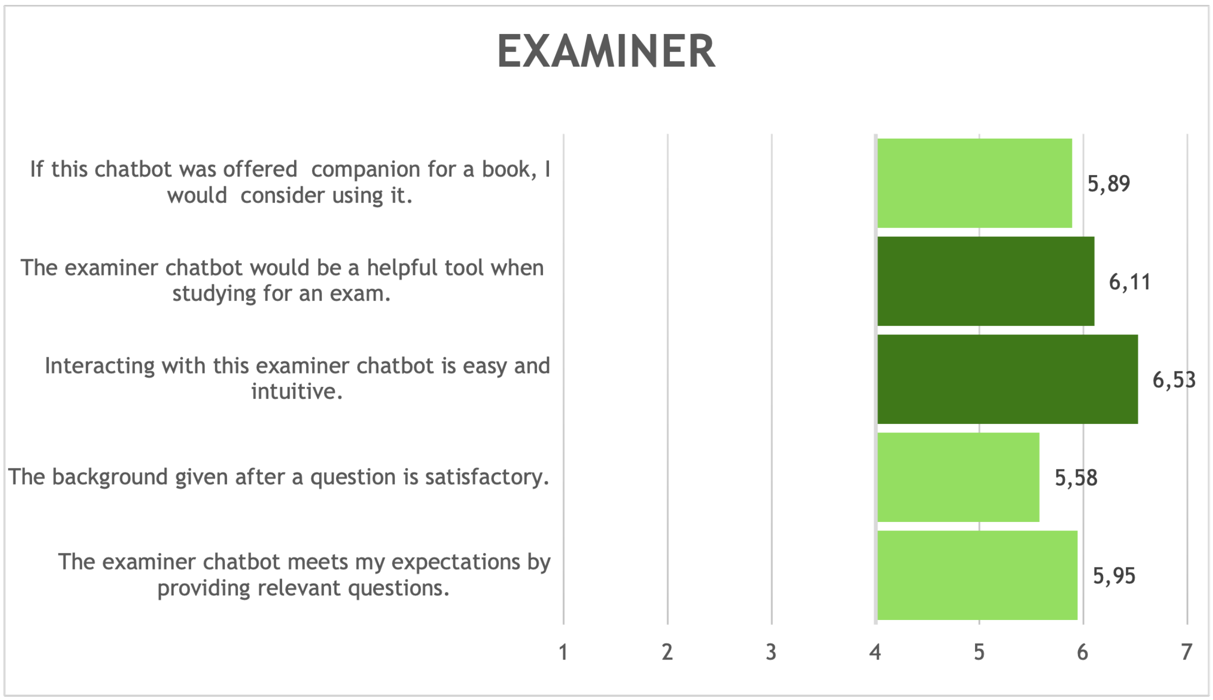


Figure 5: Average scores for the Examiner questions on a scale from *1 - Strongly disagree* to *7 - Strongly agree*

6 Discussion

The results suggest that the aims of creating a concise and correct chatbot were met. According to the user study, the tutor provides relevant responses and the examiner provides relevant questions, as well as giving a satisfactory evaluation. These findings correlate to the aim of correctness. The automatic evaluation shows our chatbot's responses are concise, compared to ChatGPT.

One improvement we can derive from the user testing of the tutor is that the testers needed to spend some time making adjustments and corrections. This usually meant rephrasing their question to get the information they wanted. This behaviour could be because when the user input is vague, the retrieval may extract another part than the intended one. This leads to the response not making sense and it is possible that perfecting the prompt could make the chatbot ask for clarification more often. Alternatively, when the similarity search is not able to find segments which are similar enough (we could define a specific threshold), the chatbot could ask a follow-up question regarding the multiple findings.

We did find that the conversation with the tutor had a natural flow, as seen in figure 4, statement 2. This could be due to the follow-up question asked by the tutor in 81 % of the responses. It creates a mixed-initiative conversation since the tutor is not merely a search tool that answers the users' questions but also engages in the conversation. According to See et al. (2019) question-asking is an attribute that improves the engaging qualities of a conversation.

Although we did not particularly experience this issue during the user testing, it would be a good idea to do the similarity search for the examiner answer evaluation in only the parts of the document that have already been retrieved when generating the question. Right now we do the similarity search in the whole chapter. Limiting the search space could make the evaluation more secure and would possibly allow the user to give shorter answers.

A challenge during the project was to develop a suitable prompt, balancing between restrictions and requirements, to create a desired behaviour. For the examiner, the challenge was to find a good harmony between giving some background to the question without giving away the answer. To restrict the LLM from giving away the answer, the final version of the prompt made the examiner sometimes skip the introduction completely and just ask a question. This trade-off was the result of the prompt being very sensitive sometimes, which we also experienced in the tutor prompt. When adding restrictions about what it was allowed to answer we lost some connection to the conversation history and when enhancing the history, it forgot about some of its restrictions. This would be necessary to explore further to get a completely safe product.

6.1 Future Work

We have identified two ways to extend the chatbot to make it usable for more people and in more ways. The first is by adding speech-to-text and text-to-speech so it is possible to talk to the bot. This would be especially relevant in the Tutor mode and one group that could greatly benefit from this functionality are younger students.

The other extension is adding math support, which would widen the subject areas the chatbot can handle. This would allow users to study subjects like physics, chemistry and math.

Another type of evaluation, that would be great to do but that does not target our stated aims, is to test if the chatbot facilitates learning. Other work in the field suggests that this is the case but it would be useful to verify that ours replicate those results.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ali, F., Choy, D., Divaharan, S., Tay, H. Y., and Chen, W. (2023). Supporting self-directed learning and self-assessment using TeacherGAIA, a generative AI chatbot application: Learning approaches and prompt engineering. *Learning: Research and Practice*, 9(2):135–147.

Alkaissi, H. and McFarlane, S. I. (2023). Artificial hallucinations in chatgpt: implications in scientific writing. *Cureus*, 15(2).

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Kuhail, M. A., Alturki, N., Alramlawi, S., and Alhejori, K. (2023). Interacting with educational chatbots: A systematic review. *Education and Information Technologies*, 28(1):973–1018.

Okonkwo, C. W. and Ade-Ibijola, A. (2021). Chatbots applications in education: A systematic review. *Computers and Education: Artificial Intelligence*, 2:100033.

OpenAI (2023). Prompt engineering. `https://platform.openai.com/docs/guides/prompt-engineering/`.

Robinson, J. D. and Persky, A. M. (2020). Developing self-directed learners. *American journal of pharmaceutical education*, 84(3):847512.

See, A., Roller, S., Kiela, D., and Weston, J. (2019). What makes a good conversation? how controllable attributes affect human judgments. *arXiv preprint arXiv:1902.08654*.

Spielvogel, J. J. (2010). *Glencoe world history*. McGraw-Hill, Columbus, Ohio.

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