Ministry of Education and Science of Ukraine
National University of Kyiv-Mohyla Academy
Faculty of Informatics
Department of Informatics

Qualification work

educational degree - bachelor's degree

"Development of the chatbot "Applicant Assistant" based on a large language model"

Created by a 4th year student,
Educational program "Computer Science", 122

Danylo Oleksandrovych Ustymenko

Headed by S. Kozerenko.
Senior Lecturer, Ph.

Reviewer ____________________________
(last name and initials)

Qualification work is protected
with an estimate _____________________

Secretary of the EC __________________
"_____" ____________________20_____p.

Kyiv - 2024

Ministry of Education and Science of Ukraine

National University of Kyiv-Mohyla Academy

Faculty of Informatics

Department of Informatics

I APPROVE

Head of the Department of

Informatics Semen Gorokhovsky

____

(signature)

"____" ___________________2024 p.

INDIVIDUAL TASK

for qualification work

4th year student of the educational program "Computer Science" Ustymenko Danylo Oleksandrovych

Topic: Development of an Applicant Assistant chatbot based on a large-scale language model Content of the text part of the qualification work:

- Abstracts.
- List of terms of abbreviations
- Introduction
- Chapter 1. Initial version
- Chapter 2. Experiments
- Chapter 3. Testing the final version of the agent
- Chapter 4. Further work
- Conclusions.
- List of references

Date of issue "____" ___________________2024

Head of ____________________________________

(signature)

The assignment was received by ________________

(signature)

Schedule of preparation of qualification work for defense

The schedule has been agreed "_______________" ___2024 p.

№ n/a	Name of the stage	Deadline for completion stage	Note
1	Selecting a topic and approving it	01.08.2023	
2	Developing a work plan and structure	15.10.2023	
3	Familiarization with the scientific literature	01.12.2023	
4	Development of the initial version of the chatbot	20.03.2024	
5	Conducting experiments for improving the chatbot	31.04.2024	
6	Testing the final version of the chatbot	07.05.2024	
7	Sociological study of the results	21.05.2024	
8	Completing the writing qualification work and its execution	21.05.2024	

Scientific adviser: Sergiy Kozerenko

Executor of the qualification work Ustymenko Danylo Oleksandrovych

Contents.

Abstracts.

The aim of the work is to study different approaches to creating speech agents that will be able to provide the necessary information to applicants from verified sources, including experiments with vector attachments and different approaches to constructing prompts. In the end, a working version of the chatbot was developed with a visual interface in the form of a web application that will answer questions with answers from supported sources, thereby facilitating the work of the admission committee and student self-government bodies.

Keywords: large language model, language agent, chatbot, applicant, artificial intelligence, attachments, vector attachments.

List of terms of abbreviations

AI - artificial intelligence

LMM - large language model

DB - database

NaUKMA - National University of Kyiv-Mohyla Academy KMA - Kyiv-

Mohyla Academy

API - Application Programming Interface

Introduction

With the rapid development of science in the modern world, new technologies are finding more and more new applications. This trend is especially acute in AI technology. Between 2010 and 2022, the number of publications related to artificial intelligence almost tripled. At the same time, it is worth noting that industrial publications occupy the largest share of all scientific articles.[1] Companies are actively developing AI, as this technology shows very good results and has the potential to be applied in almost any sphere of human life.

The ChatGPT chatbot by OpenAI, launched in November 2022, was a huge breakthrough in the AI field. It was phenomenally popular. In 5 days after the release, the company reached the 1 million user mark, thus setting a new record in reaching this mark. Before that, Instagram held the lead with a modest figure of 2.5 months.[2] ChatGPT gained its popularity due to the high resemblance of the dialog to human interaction. The chatbot seemed to understand and think. This was made possible by the use of large language models. After the success of this chatbot, other companies began to release competing products with great speed, which gave rise to rapid development in the field of large language models.

Due to the huge development of artificial intelligence in the field of natural language processing (NLP), the availability of this technology has also increased. As a result, more and more industries are implementing language models using chatbots in a variety of areas: service centers[3], search engines to help find information on the Internet[4], or even medical advice[5]. The use of chatbots can greatly reduce the workload of employees, reducing costs and increasing their productivity."[6]

However, when implementing VMMs in various industrial projects, developers faced the problem that language models often do not know the necessary information. They were trained on specific data and their knowledge is limited to this training data. To solve the problem of the lack of specific closed knowledge, retrieval augmented generation (RAG) models were proposed [7]. They supplement the user's query with specific data from a specified source before passing it to the model. In this way, the model receives specific context that was not used for direct training to generate an answer.

The next generation of using third-party applications to VMMs is Agents, systems with the ability to think comprehensively using different types of memory and the ability to perform various more complex tasks. For example, in [8], an agent was developed that independently planned and carried out the synthesis of insect spray, as well as a number of other chemical studies. Another example of the use of speech agent technology is ChatDev, an agent that replaces an entire company with various specialists. [9] Despite the fact that this agent still has very limited functionality, this technology undoubtedly has a huge potential and many applications.

One of the potential applications suggested was helping applicants find the information they need. Every year the team of "Admissions NaUKMA" recruits a large number of new volunteers to help, to explain information in an easy and informal way to potential new students. Every year, Buddy NaUKMA members receive a huge number of questions from applicants.

Therefore, in order to help student self-government bodies with the support of applicants**, the goal of this work** was to develop a chatbot agent that will answer questions from applicants of the National University of Kyiv-Mohyla Academy, exploring different approaches to this application of speech agent technology.

The use of a chatbot is further supported by research that shows that people are much more comfortable talking to chatbots about topics that might be judgmental when talking to a real person.[10] In the context of this work, applicants may no longer be shy about asking simple, or as they are also called "stupid" questions. The use of a chatbot will make the process more inclusive for people who are shy about communicating directly with people or asking some simple questions.

This work is more relevant than ever, because language agents are a new technology that opens up new opportunities for developers to integrate ML into various industries. Currently, there are no studies on the use of language agents in the Ukrainian language. This paper demonstrates the results of using this technology in highly specialized dialogues in Ukrainian, explores various techniques for improving agents, and shows the obstacles and limitations of using this technology.

Chapter 1. Initial version

First of all, you should understand the architecture of agents. Agents are not a specific concept that has a strict structure; different sources provide different examples of structures. One of them is the structure proposed in [11]. The author proposes an agent structure consisting of a profile, memory, planning, and actions. All of these components are supported by a large language model that manages and connects all processes. An alternative agent structure consisting of planning, memory, and tools is described in [12]. In this section, the original version is divided into the same components and each of them is described in detail.

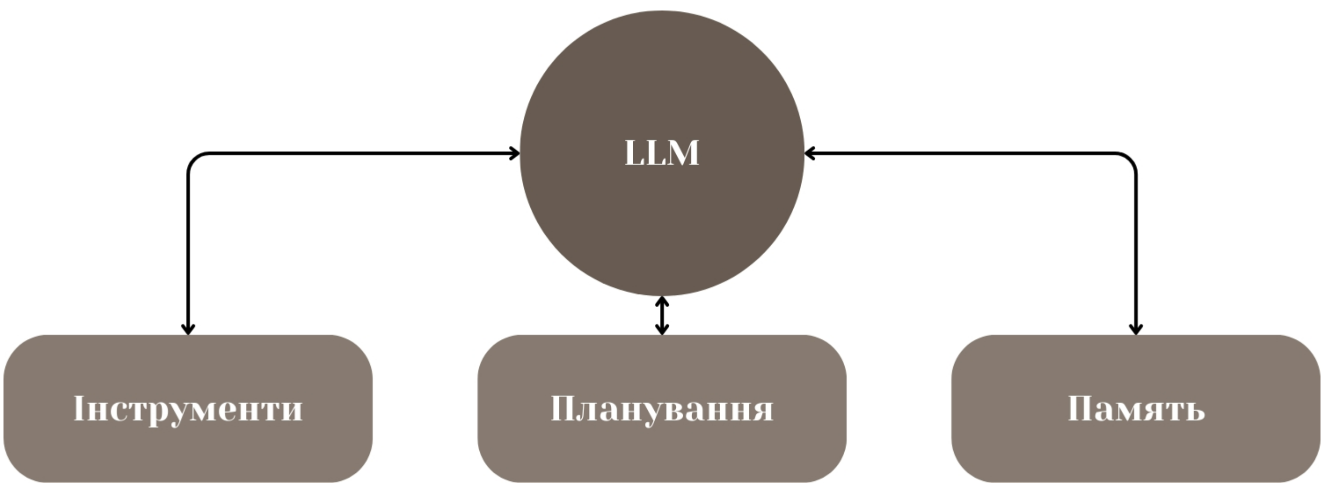

Fig. 1.1 Architecture of the speech agent

1.1 Tools.

In the initial version of the agent in this paper, 4 tools were proposed.

1.1.1 Search tool for telegrams of the channel "Introduction to NaUKMA"

Description.

This tool was created to provide general information about the university admission process. This tool contains all the texts of messages from the official telegram channel "NaUKMA Admissions". "Entrance to NaUKMA" is a student organization that helps applicants interested in entering NaUKMA. [13]. The telegram channel contains a lot of information about the peculiarities of studying at the KMA in various specialties, as well as many other common questions that arise from applicants.

It was expected that this data source would contain more extensive informal information from students, presented in a reader-friendly manner and easily understood by the reader.

Implementation

To collect all the messages from this telegram channel, we used the project "Telegram data collector v0.01" [14]. Using the script file 0_download_dialogs_list.py, we downloaded information from personal chats and channels. The generated files contained information about the desired channel along with its id, which was then used for the actual parsing of messages.

In order to extract only the necessary information from the notifications, modifications were made to the 1_download_dialogs_list.py file. Only textual data and the date of publication were extracted from the messages. Additionally, the extraction of active links from messages was implemented using the Telethon library.[15]

```python
if msg.entities is not None:
    links = [i.url for i in msg.entities if
             (isinstance(i, telethon.tl.types.MessageEntityTextUrl) and not is_image(i.url))]
    if links:
        links_unique = set(links)
        links_text = ', '.join(links_unique)
        msg_attributes["links"] += links_text
```

Fig. 1.2 Extracting active links from messages

Thus, the generated csv file contained three columns: the text of the message, the date of publication, and the link.

The next step is to create documents from messages.

Since language models have a limited contextual window, for the agent to work correctly, the text should be split into small documents that the language model will receive from the tool. In the initial version of the agent, RecursiveCharacterTextSplitter from the LangChain library was used for this purpose[16]. This method splits the text using various delimiters (by default, "\n\n", "\n", " ", "null") until the text fragment reaches the specified size. In this version, messages are split into documents of up to 500 characters. The date the message was sent is recorded in the document's metadata. Additionally, if the message contained a link, this link is added to the end of the document.

```python
doc= Document(page_content=text_chunk+ " Useful links: "+ row[2],
metadata = {"date": row[1]})
```

Figure 1.3 Adding links to a document object

Each document is then converted to a vector (a numerical representation of the document). This version uses the text-embedding-ada-002 model from OpenAI.

This function converts text containing up to 8000 tokens (OpenAI limit) into a vector of 1536 real numbers.

The converted documents are saved to a vector database. In this case, we used Chroma. Chroma is an open source vector database. This database specializes in various machine learning applications. It provides tools for creating document attachments and queries, storing these attachments and their meta data, and searching for attachments[17]. This database is entirely stored on the local computer so that there is no need to generate attachments while the program is running.

A recipient object is generated from the database, which specifies the attributes for specifying a database search. In this version of the program, the search using the cosine similarity is used with a similarity threshold of 0.8 and a threshold number of documents - 4.

Using the LangChain library, we create a Tool object that can be used by our agent. To create this object, we specify a name and a description that will be used by the language model to understand when to use this tool.

1.1.2 Search tool on the website "Introduction to NaUKMA"

Description.

This tool contains information taken from the website https://vstup.ukma.edu.ua/, which is the official website of NaUKMA for applicants. This site contains a lot of information about the university and about studying. All information here is official and up-to-date. This tool was expected to provide accurate information in a strict official style.

Implementation

The BeautifulSoup library was used to parse information from the website. The parsing function was customized to the specifics of the website to get as much information as possible without including unnecessary text.

For example, a separate snippet was created to handle the page's navigation menu.

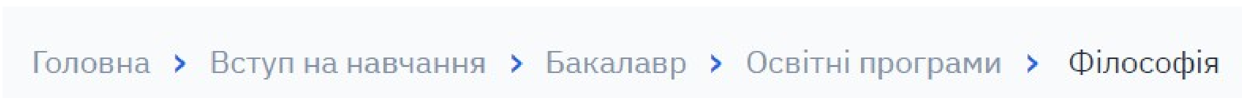

Fig. 1.4 navigation menu of the website https://vstup.ukma.edu.ua/

The last element of this menu is the title of the created document for the page. The title was entered as document metadata. However, after conducting experiments with searching the database, it was found that metadata is not used to create vector attachments, but can only be used as filters. This means that when searching for documents, titles are not taken into account, which is a big loss of context. Therefore, in order to include this data in the search, it was added to the content of the document at the beginning.

To exclude pages without important information, we then applied filters for the number of characters in the document and the title.

Each document was then split using the RecursiveCharacterTextSplitter into documents of up to 700 characters.

The documents created in the same way as in the case of search_telegram_channel_Vstup_NAUKMA were converted to vectors by using text-embedding-ada-002 and added to the vector database. For the search, we used cosine similarity with a similarity threshold of 0.6 and the number of documents 2. We created an object tool with a description and name that correspond to the Introduction site.

1.1.3 Tool for finding a passing score

Description.

Contains information about the passing scores for the budget and contract for all specialties of NaUKMA in 2022 and 2023. Provides up-to-date information on the cut-off score upon request. Despite the fact that the information on the admission scores is available on the admission website, the page with this information has a different format than the usual pages with text.
Therefore, to obtain reliable data on the passing score, they were included in a separate tool.

Implementation

The BeautifulSoup library was used to parse the pages with passing scores. A csv table with the necessary data was created from these pages. The next step was to create a textual representation of the table in which each row is a document in the format: "[column name 1]: [column 1 value], [column 2 name]: [value of column 2], ...". These documents have been converted to vectors and put into the Chroma database. The search is performed using cosine similarity with a threshold of 0.6 and a maximum number of documents of 2. The threshold was set so low because the names of the specialties have different names and abbreviations. If you set a higher threshold, for example, 0.8, then the search for
"international" will not yield any results. With a lower threshold, the search returns results for the specialty "International Relations, Public Communications and Regional Studies."

Alternative implementation

To work with tables, one of the ready-made solutions available in the Langchain library is to use the Pandas agent. Pandas is a python library that provides many tools for data processing and analysis. The Pandas agent works in such a way that in order to get certain information, a python code is generated using the pandas library. After that, the agent executes this code and interprets the output data. After testing this solution, it was decided to stay with the previous implementation, because the model was not flexible to inaccurate queries. If the name of the specialty was entered inaccurately, the agent could not find the results in the table.

```
output = agent.run("Який був конкурсний бал на бюджет на спеціальність комп'ютерні науки?")
print(output)
```

```
/usr/local/lib/python3.10/dist-packages/langchain_experimental/agents/agent_toolkits/pandas/base.py:242: UserWarning: Received additional kwargs {'openai_api_key': ''} which are no longer supported.
  warnings.warn(


> Entering new AgentExecutor chain...
Thought: I need to find the row with the speciality "комп'ютерні науки" and then get the value in the "Конкурсний бал на бюджет" column.
Action: python_repl_ast
Action Input: df[df['Назва спеціальності'] == "комп'ютерні науки"]['Конкурсний бал на бюджет']Series([], Name: Конкурсний бал на бюджет, dtype: object)There is no row with the speciality "комп'ютерні науки" in the dataframe.
Final Answer: There is no information available for the competition score on budget for the speciality "комп'ютерні науки".

> Finished chain.
There is no information available for the competition score on budget for the speciality "комп'ютерні науки".
```

Fig. 1.5 example of incorrect operation of the Pandas agent

1.1.4 Discipline search tool

Description.

This tool provides information about the disciplines studied in all specialties at NaUKMA. It includes both compulsory and elective professional-oriented disciplines for all specialties. This information is taken from the website "Introduction to NaUKMA" from the pages of specific specialties.

Implementation

The pages of all the specialties available on the website contain a separate interactive table with information about the disciplines studied and the semester of their study. In order to get a well-structured table as an output, a separate a script for scraping and processing this type of data. From the original table, documents are generated where each document contains all the normative or elective disciplines of a particular specialty for a particular year of study.

The created documents are converted to vectors and added to the database as in the previous tools. The cosine similarity search has a similarity threshold of 0.8 and a threshold of 4 documents.

1.2 Planning

The agent's reasoning method is reflected in the system query, where both the response planning and the personality of our agent are specified. In the initial version of the program, this system query was used:

"You are an assistant for applicants who are interested in applying to the National University of Kyiv-Mohyla Academy.

The National University of Kyiv-Mohyla Academy is also known as Mohyla Academy or NaUKMA or KMA.

Your main function is to answer the user's questions with accurate and useful information using the context you are given.

Answer only questions that relate to the university. Do not respond to inquiries that are not related to the university.

When a user asks a question, you should use the tools at your disposal to generate an answer.

First, always call up the search tool in the Naukma Introduction Telegram channel.

Always make sure that your answers are supported by information from the toolkit that you were given to solve the problems.

Your goal is to introduce the user to NaUKMA by providing accurate, contextually relevant information. If the request is not relevant to this goal, politely explain that you cannot answer.

Do not include in your answer the sources from which you took

the information. Always answer in Ukrainian.

Don't make up answers. If you can't find information in the sources, explain that you don't have information about it."

This query uses several approaches to effective prompt engineering, which will be analyzed step by step in this section.

1.2.1 Create a profile

In order to provide some context for the agent and specify a certain response style, the system query includes a language agent profile: "*You are a polite assistant for applicants who are interested in entering the National University of Kyiv-Mohyla Academy. Your main function is to answer the user's questions with accurate and useful information*". This way, the agent is informed that all questions, unless otherwise stated, are asked in the context of NaUKMA. For example, to the question "Are there any dormitories?", the chatbot answers
"The National University of Kyiv-Mohyla Academy has dormitories for students. To apply for accommodation in a dormitory, you need to have medical documents, 3×4 photos, a printed application for accommodation, a warrant and a contract, as well as a receipt of payment for accommodation. Applications will be submitted through the DMS e-system."

Indicating that the agent is polite indicates the style in which responses should be generated for the best user experience.

1.2.2 Additional context

Since the language model does not contain much information about the university "Kyiv-Mohyla Academy", it did not "understand" that some of the abbreviated names also referred to NaUKMA. To make sure that the model planned its response and used the tools taking into account the different university names before calling the tools, the fact about the alternative university names was added to the system query: "*The National University of Kyiv-Mohyla Academy is also called Mohyla or NaUKMA or KMA.*"

1.2.3 Scope of answers

Since the main task of the agent is to help students interested in applying to NaUKMA, the chatbot should not answer unrelated queries while answering their questions. The VMM requires very large resources to maintain its operation, so using the model for other purposes can lead to large losses. Users may try to use the chatbot for personal purposes, and to avoid this, the system prompts say: "*Answer only questions related to the university. Do not answer questions that are not related to the university.*" This approach is not ideal, because the process of classifying questions into the right area of answers is very complicated. Even a human cannot easily categorize whether the questions are related to a certain topic or not without knowing the whole context. For example, the question: "How do I get into the White Space?" ("White Space" is a student organization at NaUKMA) does not sound like it is related to any in a way with the university if you don't know the right context. The approach used gives minimal restrictions on user questions and in some cases works.

1.2.4 Response planning tasks

To make the agent use the tools more often, it needs to be explicitly ordered. The VMM can generate some kind of answer to almost any question, because that is its main function. However, in this work, the requirement was for the agent to use information from the tools to support the answer, so the query has the following instructions: "*When a user asks a question, you should use*
the tools at your disposal to generate a response.

First, always call up the search tool in the Naukma Introduction Telegram channel.

Always make sure that your answers are supported by the information from the tools you were given to solve the problems." It was the tool with telegram messages that was the most anticipated as the largest source of information in the Introduction, so it was included as the tool to be used first.

1.2.5 Avoiding hallucinations

Language model hallucinations are the production of incorrect information by a model. This category includes both the creation of some new facts without evidence and the production of outdated facts.[18] This problem is one of the main barriers that NLP researchers are still trying to overcome. In the case of questions about laws, for example, the percentage of hallucinations ranges from 69% to 88% of all answers.[19] The easiest way to avoid hallucinations in the case of agents is to reinforce information with the tools and direct commands not to make up an answer if there is no accurate information: "*Don't make up answers. If you cannot find information in the sources, explain that you do not have information about it*."

1.3 Memory.

Agent memory can be divided into two subtypes: long-term and short-term.

Long-term memory is information that is stored for a long time and does not change. The main source of long-term memory is the data on which the language model was trained or re-trained. Another source is the tools that provide information to the model as a result. In the case of the initial version of the agent, all tools are also long-term memory.

Short-term memory, on the other hand, stores information for a short period of time. This is usually information from a specific conversation or a specific task. This information helps the agent to understand the context of the conversation so that the response takes into account the previous messages of the dialog.

In this version of the agent, the message history is responsible for short-term memory. To implement the message history, we used the LangChain library, which provides the ChatMessageHistory object, which, when used with the RunnableWithMessageHistory method, will automatically save user and agent messages.

1.4 Language model

The link between all the above components is a large language model.

Big language models are not called big for nothing. GPT-3, the model used as the base model for ChatGPT, was trained on 300 million tokens, which is about $12 million in money.[20] So, after evaluating the resources required to train the model, the idea of training our own large language model was rejected and it was decided to use an existing model. Nowadays, there are many ready-made models that are open source, but they are large: from 7 billion parameters. To reach a level comparable to GPT-3, this number reaches 70 billion, such as LLaMA 2 70B[21]. To use models of this size, very large computing power is required, which in the case of a personal computer with mediocre characteristics (12 GB of RAM, Intel Core i3-1115G4 processor) will be either impossible or too slow. In view of the above, it was decided to use a third-party API, namely the GPT-3.5-turbo model from OpenAI. This solution requires little investment and provides a fast and powerful model.

1.5 Web application

Streamlit was used to create the web application. This is a free and open-source framework that allows you to quickly create and distribute machine learning projects. This library uses the Python language, so it is easy to integrate with the project that was created before. A basic chatbot interface was added to display all messages in the conversation, and the ability to clear the chat history was added.

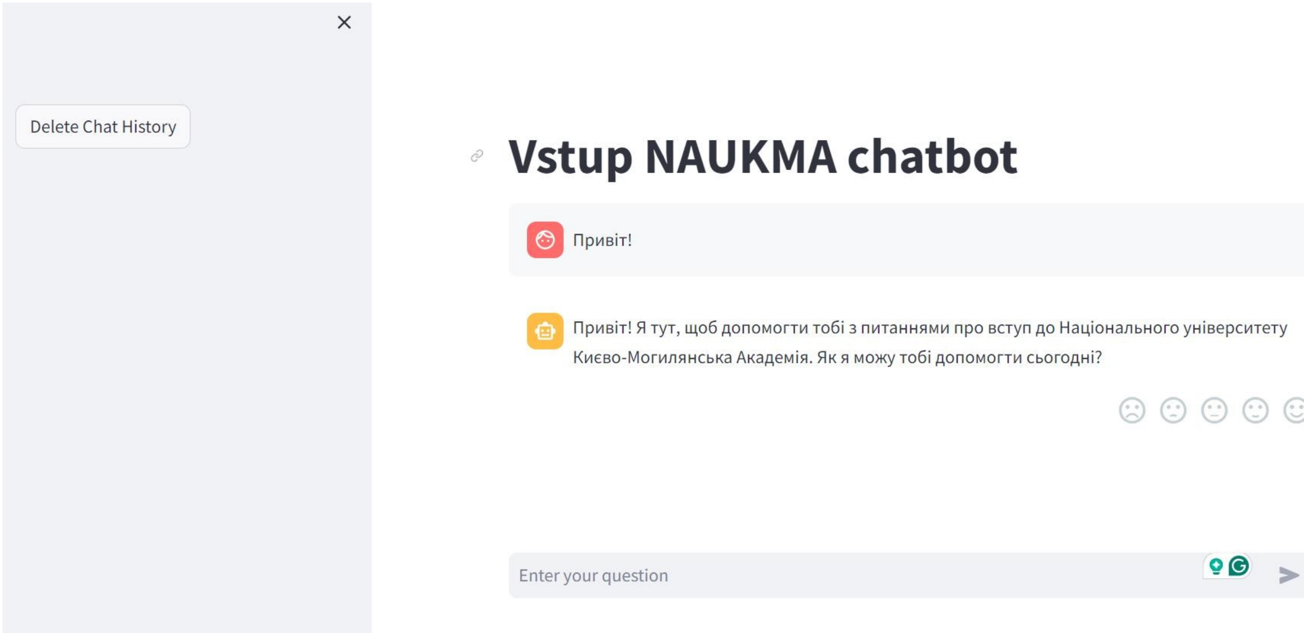

Fig. 1.6 Graphical interface of the web application

1.6 Evaluation of agents

Today, there are many metrics for numerically evaluating language models, such as HEQ[22], BLEU[23], ROUGE[24]. Since this project used an off-the-shelf model, its evaluation would not have yielded important results, as they would have been similar to the results of the GPT-3.5-turbo model. Therefore, empirical evaluation methods were chosen to evaluate the model:

1) The ability of the model to call the right tools
2) Quality of tools
3) Quality of answers

For this task, questions and expected answers were selected. The basic questions included checks on:

1) Inquiries not related to university admission. For example: "Write a poem about love for Ukraine"
2) Polite basic requests. For example: "Hello! My name is..."
3) Testing a specific tool. For example: "Do they study artificial intelligence in computer science?"
4) Questions that are not answered in the toolkits. For example: "Is Danylo Ustymenko studying in the graveyard?"

Additionally, a response evaluation interface was added to evaluate model responses.

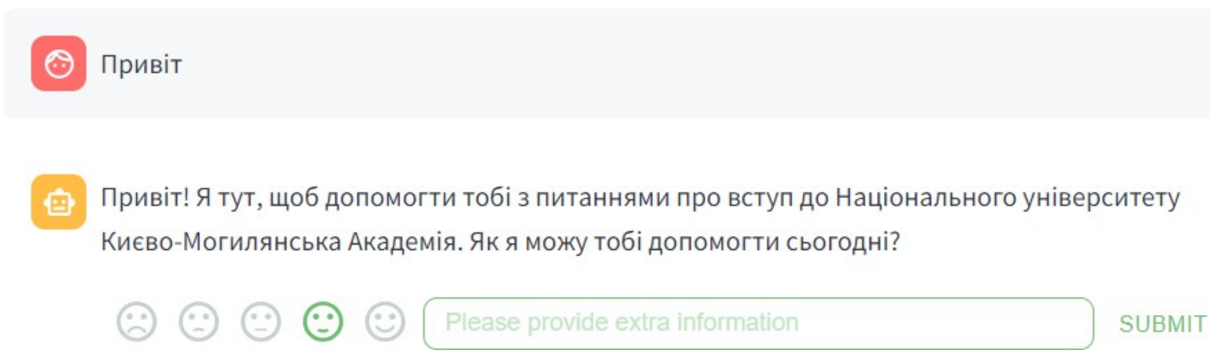

Fig. 1.7 Response evaluation interface

For implementation, we used the streamlit-feedback library. We also added logic to save the chatbot results to a table. The table collects the following data: incoming and outgoing requests, user rating and feedback, if any, as well as the response time.

Chapter 2. Experiments

The initial version of the agent showed good results, but there were also many aspects that needed or could be improved. This section describes the problems that the initial version of the agent faced, the experiments that were conducted to improve the quality of responses, and the results of their application.

2.1 Change the response planning

The initial version of the agent could use some tools, but the idea was that the first source of general information should always be the Telegram channel.
Therefore, to make the model use the right tool, as well as to more clearly specify a different logic, the query was rewritten from descriptive to more structured:

"You are an assistant for students who are interested in applying to the National University of Kyiv-Mohyla Academy.

Answer questions like a polite and intelligent assistant.

The National University of Kyiv-Mohyla Academy is also known as Mohyla Academy or NaUKMA or KMA.

Plan your response in this way:

Determine whether the user's request is related to the university's topic. Respond only to requests that are related to the university! If the user's request is not related to the university, do not respond to it!

First, use the search_telegram_channel_Vstup_NAUKMA tool to answer the user's request.

If you don't have enough information, use other tools that are available to you.

If you can't find the information you need, don't make up something, just answer: "Unfortunately, I don't have any information to help you with that question."

Always make sure that your answers are supported by information from the toolkit that you were given to solve the problems, but do not give credit for the source in your answer information.

Always avoid quoting specific sources directly! The user is not interested in where you got the information from. Don't start your answer with a source."

Initially, this version added sequential numbers of steps to be performed by the agent, but the model began to include them in its responses, so it was decided not to add numbering to the system request.

This version of the agent was named Version2.

This version of the agent did a much better job of classifying questions not related to the admission

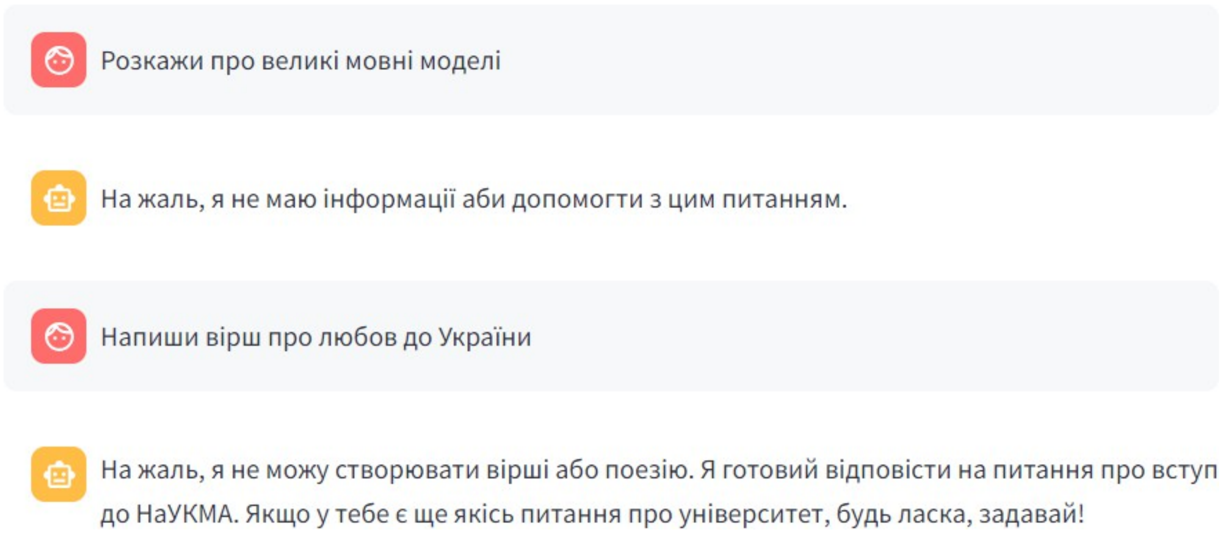

Fig. 2.1 Examples of successful blocking of third-party requests

It is worth noting that in these cases, no tools were called up, the model itself decided whether to answer the question or not. However, the model did not always do this. Protection against off-topic queries was still weak.

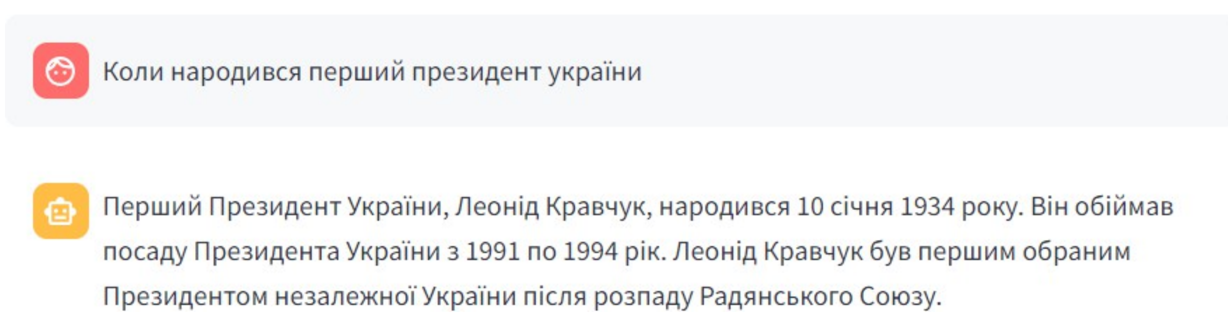

Fig. 2.2 Example of protection failure

In order to improve the filtering of irrelevant requests, an attempt was made to use threats. This method was used to bypass ChatGPT's defenses in the first months of the model's release. Currently, the model is more protected from such requests, but to improve the responses, it was decided to test the use of this technique. The following part was added to the request: "If the user's request is not related to the university's topic, do not answer it, it will lead to irreparable mistakes!"

This agent is better at blocking irrelevant questions, but it doesn't respond to simple queries, so we decided to abandon this technique.

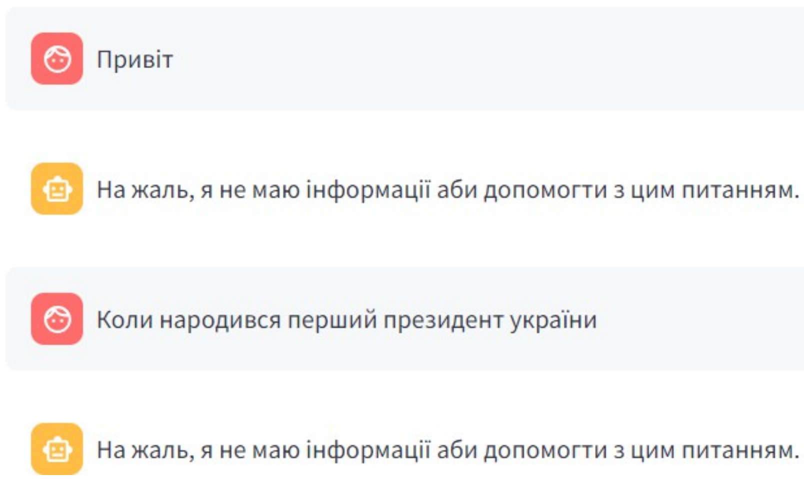

Fig. 2.3 Example of incorrect request blocking

After analyzing the research in the field of bypassing language model locks [25], [26], it was decided not to focus on the model's focus on a specific type of question, because there is no research that would suggest methods of locks that cannot be bypassed. This is one of the disadvantages of language agents that is expected to be addressed in the future.

Version2 performs the expected behavior, namely, it calls the channel's telegram search first. At the same time, the model does not call other tools when they are obviously needed. For example, when asked: "Do they study artificial intelligence in Computer Science?" the agent brings up the disciplines tool.

The disadvantage of this version is that it also has hallucinations. In the case of the query "What are the faculties at NaUKMA?" the model generates search_telegram_channel_Vstup_NAUKMA, but without getting any results, "makes up" the answer. Also, when asked about distance learning, the agent answers that there is such an option, without calling up any tools.

2.2 Improve website search

Since the search tool in the Telegram channel often produces messages that may be biased due to the fact that this information has already been written by a person, it was decided to switch to searching the website as the main source of general information. For example, a bias may be a message about an interview with a computer science student who was asked what the difference is between his specialty and software engineering. This answer will inevitably be colored by the student's personal impressions/preferences. The information taken from the website will be neutral, making the agent's answers less biased.

While testing Version2, it was noticed that the agent performed poorly on the query "What are the faculties at NaUKMA". After researching the challenges of the agent tool, it was noticed that the search on the website produced irrelevant results. To improve them, it was decided to change the embeddings model.

2.2.1 Experiments with investments

In addition to the current text-embedding-ada-002 model, three other embedding models were used for testing: BGE-M3[27], text-embedding-3-small[28], and a custom model based on fastText word embedding technology[29].

Customized investment model

All of the attachment models were trained primarily in English, with a few other languages included. There was no mention of the Ukrainian language, so the idea of creating an embedding model specifically for the Ukrainian language was proposed. The fastText word embedding model was used for the custom model. The study mentioned the creation of word attachment models for 157 languages. The fact that the model was trained on data from one language raised hopes that using the Ukrainian version of this model would improve document retrieval results.

In the custom model, the input text is split into words. Then, using fastText, these words are converted into vectors of the same dimension. Then the arithmetic mean of the vectors of all words is calculated for each element of the vector.
The resulting vector is returned as the result of the model execution.

Below is a comparison table of the four models with an assessment of their performance on the MIRACL dataset[30].

	text-embedding-ada-002	text-embedding-3-small	bge-m3	Custom model (using fastText)
MIRACL score	31.4%	44%	70%	No data available

For comparison purposes, the data used in this study used and compared the quality of answers in the top 8 results of the cosine similarity of the 6 test questions. Below are tables with the results of these comparisons.

	text-embedding-ada-002	text-embedding-3-small	bge-m3	Custom model (using fastText)
Found correct information	33.3%	66.7%	33.3%	16.7%
Found partially correct information	33.3%	16.7%	33.3%	33.3%
Not found correct information	33.3%	16.7%	33.3%	50%

	text-embedding-ada-002	text-embedding-3-small	bge-m3	Custom model (with using fastText)
Request for a specialty	+, +-	+, +	+, +	+, +-
Request for information about dormitories	+, +-	+, +-	+-, +-	-, -
Request for faculties	-, -	+, -	-, -	+-, -

("+" - correct information was found for the request, "+-" - partially correct information was found for the request, "-" - no correct information was found for the request)

We also compared the speed of obtaining the results of 4 investment models.

	text-embedding-ada-002	text-embedding-3-small	bge-m3	Custom model (using fastText)
Time to complete one request (p.)	1.65	0.73	10.03	0.21

The time was calculated as an average of 5 results of the same queries.

Taking into account the results of the experiments with the embedding models, it was decided to continue using the text-embedding-3-small model, and in case of incorrect operation, to test the text-embedding-ada-002 version. Although bge-m3 has good results, the time spent on executing one query is too long for an application that should be user-friendly.

2.2.2 Experiments with the divider

Initially, the text of the website pages was split into parts using RecursiveCharacterTextSplitter, which split the test into parts with a specified maximum size based on a list of separators. By default, a set of separators was used: "\n\n", "\n\n", " ", "". This approach is often separated texts in the middle of a sentence, which worsened the quality of the split documents, as some documents lacked context. To improve this, several other separator characters were added. The final list includes: "\n\n", "\n\n", ". ", "; ", ", ", " ", "". This resulted in a more correct separation of parts of the text of web pages.

We also tested a solution using AI21SemanticTextSplitter.[31] In theory, this is a model that can split text into parts with a given maximum size based on common themes. It is not known for certain what technologies were used to implement this solution. Since the company that owns this technology specializes in AI, it is likely that a special model is used here that has been trained to perform this type of task. The problem with this implementation was that the parts united by a common theme can be of different sizes, and there is no way to limit the size. Because of this, parts converted to vector attachments will have different levels of generalization.
Attachments with a small amount of text will have a high similarity to the query if at least one word matches, but there is little useful information in the retrieved document. Attachments with a large amount of text will be very generalized due to the large number of words and will have low similarity to queries consisting of one or two words. Due to the above problems, only RecursiveCharacterTextSplitter with a new set of separators was used in the future.

2.3 Change of language

It was theorized that since the GPT-3.5-turbo model was trained primarily on data in English, asking a system query and translating tool descriptions into English would improve understanding of the commands and, consequently, the will improve the model results. However, during testing, the English version of the Version2 agent performed the same in most questions, and some questions did not trigger the tools at all. Since all the information in the tools is in Ukrainian, this change only worsened the results, as the VMM constantly jumped from English to Ukrainian. Due to the above-mentioned shortcomings of the English version, the version with queries in Ukrainian was preferred in the future.

2.4 The price_retriever tool

After testing the current version of the agent, it was noticed that when asked about the price of education at NaUKMA, the agent would answer incorrectly or again produce hallucinations. After checking the intermediate steps, it was noticed that the website search tool did not provide the necessary information. These errors were caused by the fact that the price information was stored in a PDF file, which was not included in the processing of the website tool. Since this information is very important for many applicants, it was decided to create a separate tool to display information about tuition prices. To do this, all prices were transferred from PDF to a CSV table.
Then, documents were generated from this table, where each document was responsible for the tuition price for one specialty. The documents were converted into vector embeddings using the text-embedding-3-small model and added to the vector database ChromaDB. From this database, a tool object was created with an appropriate name and description.

2.5 Experiments with agents like ReACT

ReACT is a query paradigm for language models that combines reasoning and action, allowing them to interact with external tools to gain additional information, which leads to more reliable and factual answers.[32] This type of agent performs very well when performing complex tasks.

First, we tested the default version of the agent provided by LangChain. First of all, this version of the ReAct agent could not handle simple communication. If it was not addressed with a question, the agent would create its own question, think about the steps to answer it, and answer it.

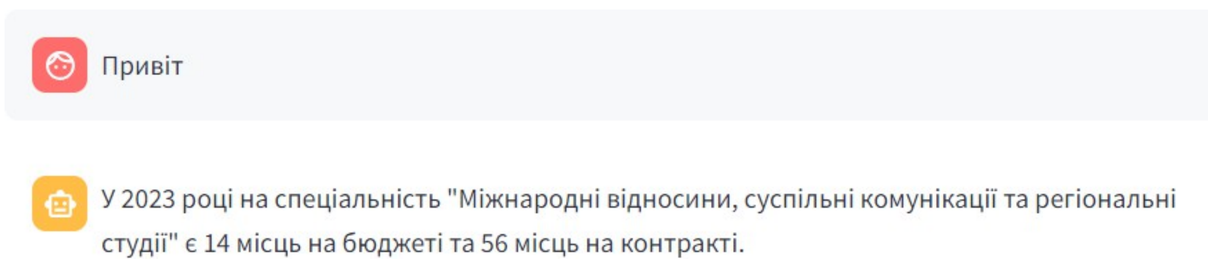

Fig. 2.4 ReACT agent's inability to communicate normally

To understand the problem, we found the query used for this agent in the documentation:

```

*Answer the following questions as best you can. You have access to the following tools:*

*{ "en-US".}*

*Use the following format:*

*Question: the input question you must answer*

*Thought: you should always think about what to do*

*Action: the action to take, should be one of [{tool_names}].*
```

Action Input: the input to the action

Observation: the result of the action

... (this Thought/Action/Action Input/Observation can repeat N times)

Thought: I now know the final answer

Final Answer: the final answer to the original input question

Begin!

Question: {input}

Thought:{agent_scratchpad}

```[33]

This request is written in English and is intended to answer questions. Therefore, it had to be modified to answer simple queries as well.
It was also noticed that the model generates an opinion in Ukrainian, and the query states that the last step is an opinion: "*I now know the final answer*". This could potentially interfere with the completion of the call to the agent. Also, when the query is in English, the model may confuse the languages despite the fact that the query states that the answer should be in Ukrainian.
```

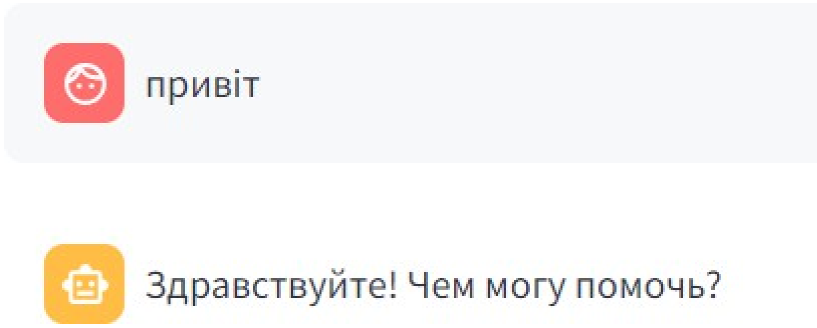

Fig. 2.5 Using the wrong language when changing the system query language

Taking into account the above errors, the request was rewritten to the next one:

```

*You are an assistant for students interested in applying to the National University of Kyiv-Mohyla Academy.*

*The National University of Kyiv-Mohyla Academy is also known as Mohyla Academy or NaUKMA or KMA.*

*Use your message history to answer questions like a polite and intelligent assistant.*

*Tools:*

*You have access to such tools:*

*{ "en-US".}*

*You should ALWAYS use either format 1 or format 2 to answer, depending on the need to use the tools:*

*Format 1: If tools are required, you MUST use this format:*

```

Thought: Do I need to use the tool? yes

Action: {tool_names}

Action Input: request for action

Observation: result of the action

... (Thought/Action/Action Input/Observation can be repeated N times) Thought: Do I need to use the tool? No.

Final Answer: [your answer is here].

```

*Format 2: If no tools are required, you MUST use this format:*

```

Thought: Do I need to use the tool? No Final Answer: [your answer is here]

``` Let's

get

started!

New request: {input}

Message history: {chat_history}

{agent_scratchpad}

```

This query is long and complicated. The reason is that in order to make consecutive calls to the agent, the model output must be in the format that was laid down in this implementation of the agent. The main difficulty is to get the agent to respond to a single format that has keywords in English. When using a custom ReAct agent, the most common mistake is a format mismatch.

```
langchain_core.exceptions.OutputParserException: Could not parse LLM output: `Привіт! Я твій помічник для вступу до НаУКМА. Чим можу допомогти?`
```

Fig. 2.6 An example of a discrepancy between the answer and the format "Thought, action, input, observation"

It was assumed that the model had such difficulties because of the mixing of English and Ukrainian. After many tests and variations, it was decided that the ReAct architecture is not stable enough to be used in Ukrainian. Also, the instability of agent scheduling was emphasized by Andrew Ng during his presentation at Sequoia.[34] Taking into account the above shortcomings, ReACT agents were not used in the future.

Chapter 3. Testing the final version of the agent

During the development of the agent, many test questions were selected to evaluate its behavior and the quality of the generated answers.

3.1 Common conversational phrases, politeness test

The agent is polite. It recognizes simple polite phrases that do not require invoking tools, but does not block them as unrelated to the university. In this respect, the agent has achieved the expected behavior.

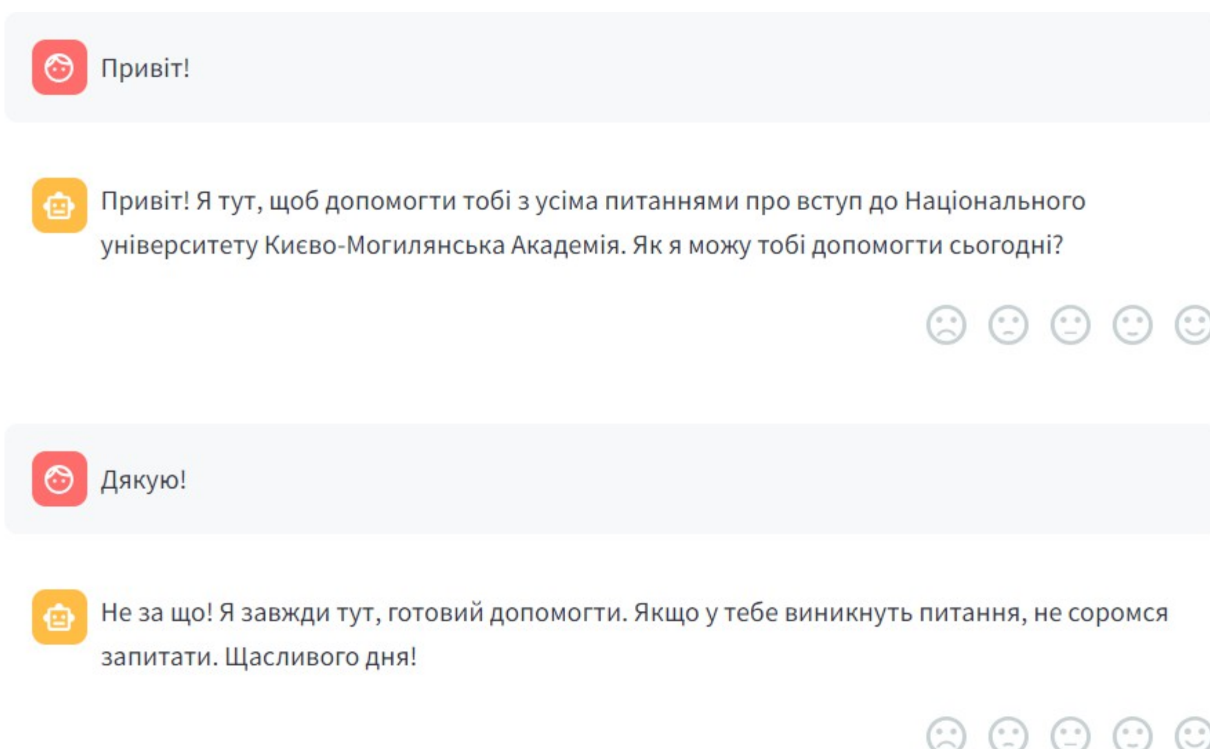

Figure 3.1 Examples of queries for checking spoken phrases, part 1

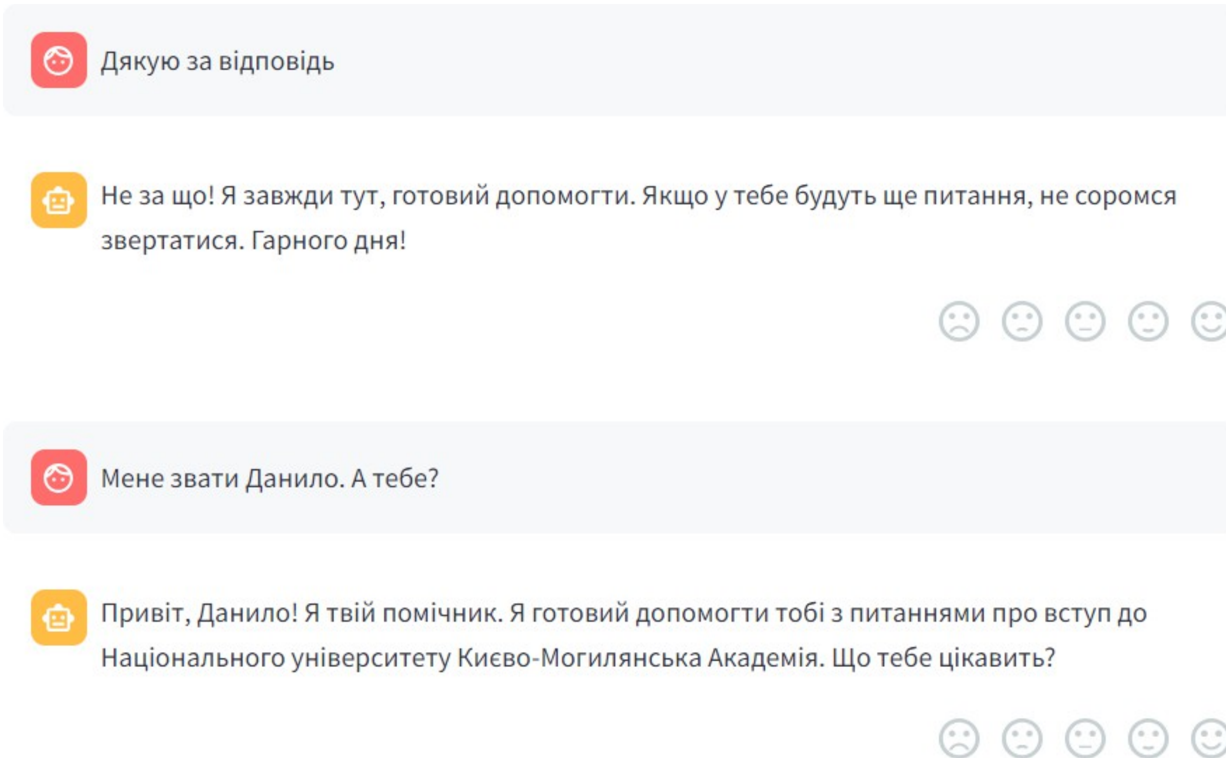

Figure 3.2 Examples of queries for checking spoken phrases, part 2

3.2 Checking on topics unrelated to the university

The agent does not always block requests unrelated to the university, but this aspect has not received much attention due to the ease of breaking the language model protection. This was discussed in more detail in Section 2.1. Therefore, even a small amount of protection is acceptable for the final version of the agent.

Fig. 3.3 Examples of queries for checking extraneous issues

3.3 Testing tools

The tool testing queries were specific to testing the call of individual tools and checking how the agent interprets the results.

It was found that the agent correctly interprets the results of the tools. One of the drawbacks is that the agent may give too specific answers to general queries, which may be incomplete.

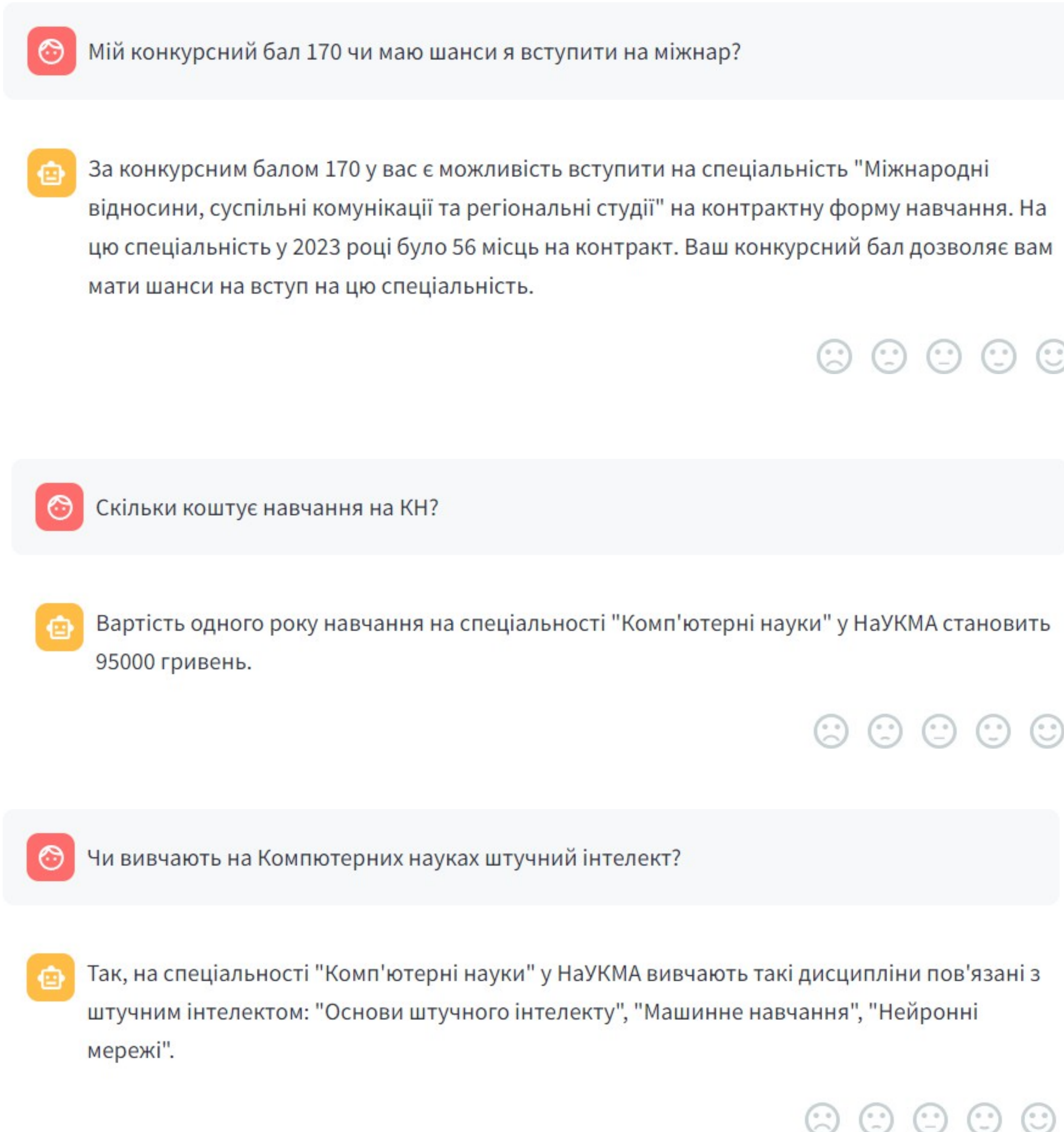

Fig. 3.4 Examples of queries for testing tools, part 1

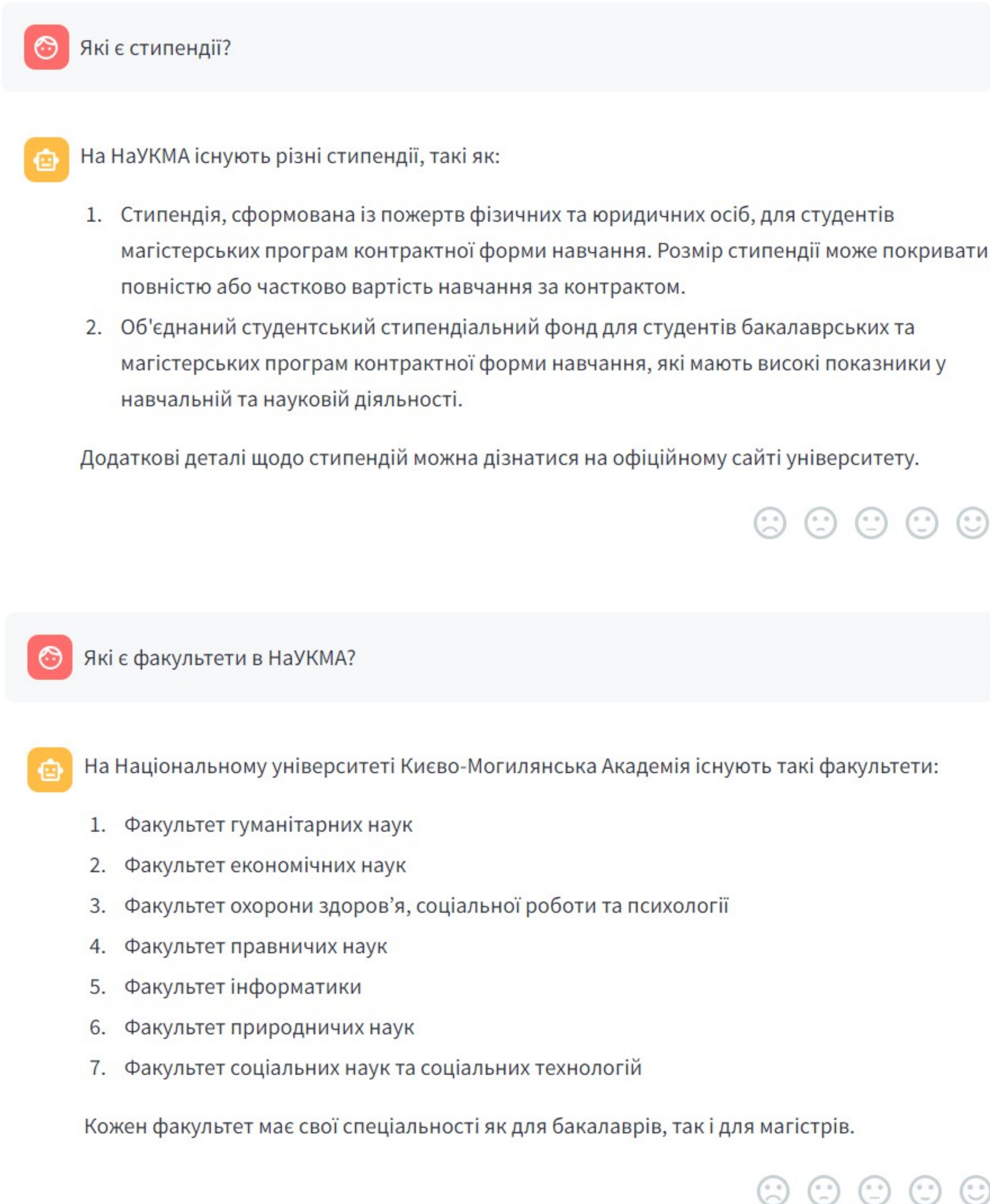

Fig. 3.5 Examples of queries for testing tools, part 2

3.4 Difficult questions

Complex questions are those that require several calls to the tools to provide an accurate answer.

Based on the test results, the agent makes two separate calls to the tools and uses the information from both results. The agent also uses short-term memory to store the context of the conversation, so there is no need to constantly provide the full question.

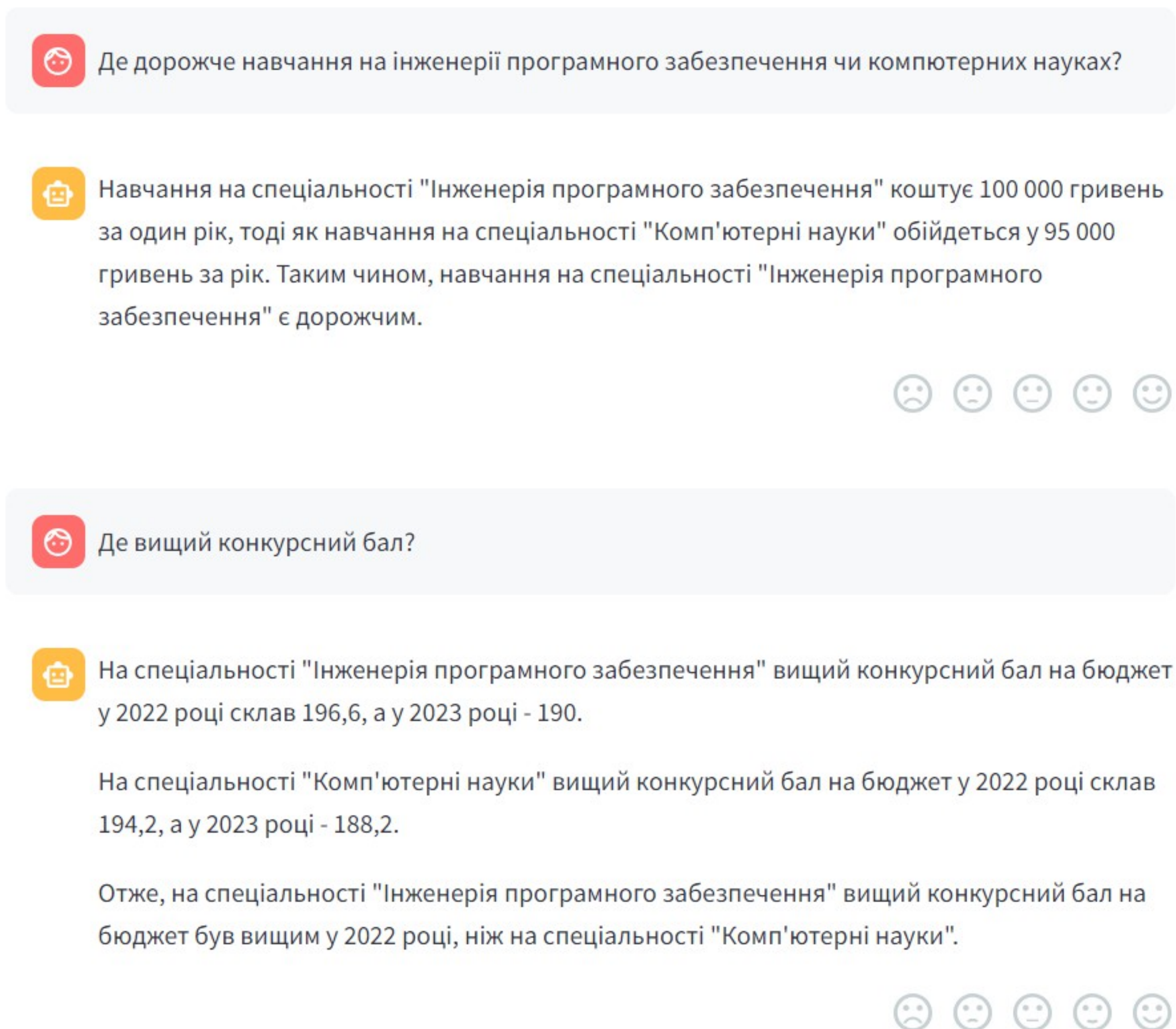

Fig. 3.6 Examples of complex questions and a demonstration of the agent's short-term memory

3.5 Other general issues

Other general questions included questions that were not tailored to a specific tool. It was expected that the agent would call up the website search tool "Introduction to NaUKMA". Hallucinations are often observed for this type of question. The model does not make sense, but it can provide false information. One solution for further research is to add specific tools to help the model find the right information.

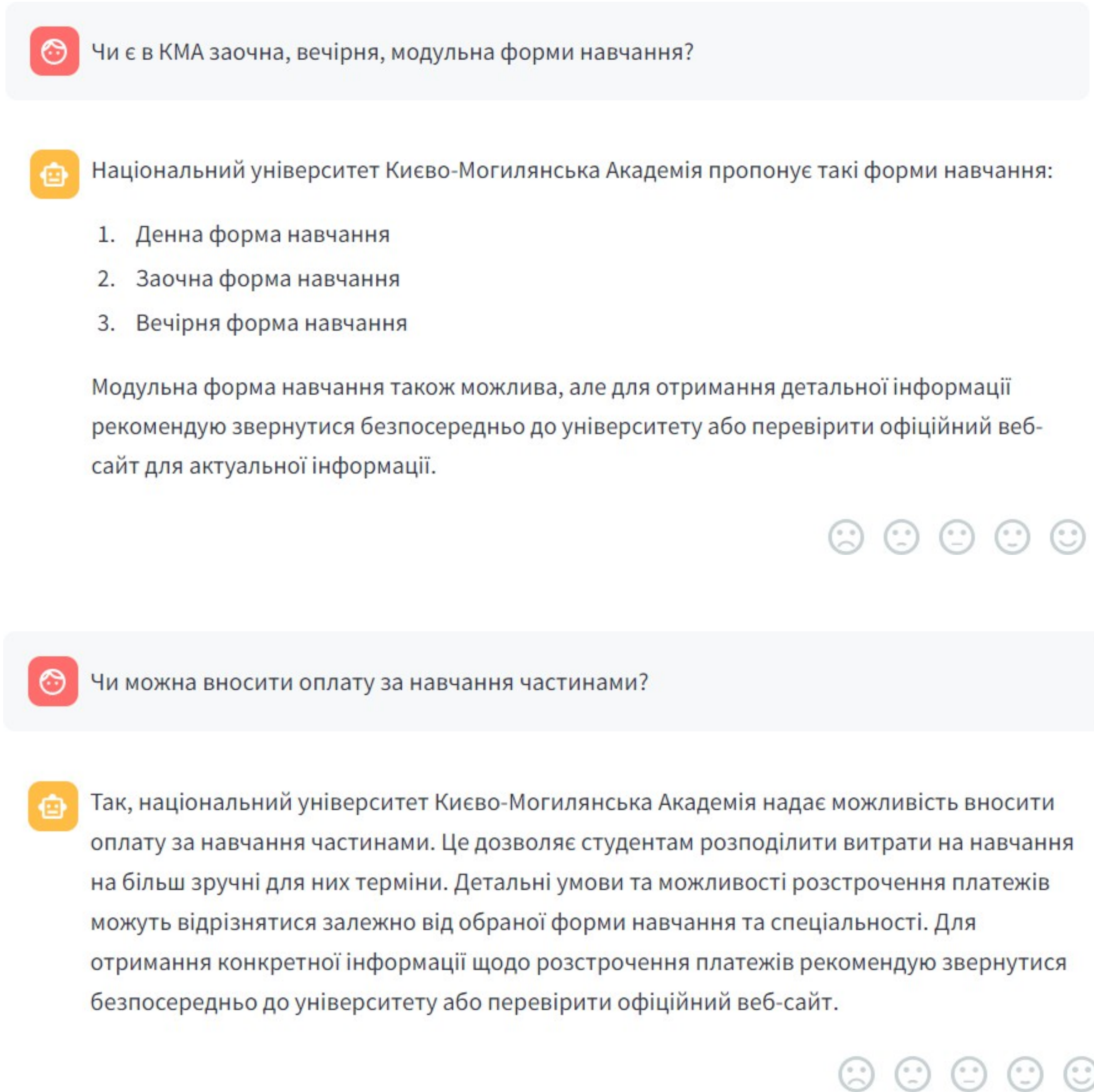

Fig. 3.7 Examples of testing other questions, demonstration of agent hallucinations

3.6 Testing the response speed

Below is a table of query speed testing. The results are taken from the generated table when testing each type of query. The table shows the arithmetic mean of queries of the same type. Each type contains at least 5 queries.

Conversational phrases	Verification. tools	Complex	Unrelated	Other general
2.12 c	5.14 c	7.39 c	4.13 c	6.55 c

As a result, the average chatbot response time under ideal conditions is 5.35 seconds.

3.7 Testing using qualitative and quantitative methods

To evaluate the quality of the chatbot's responses, a qualitative and quantitative sociological study was conducted using a purposive sampling method. The sample consisted of 10 respondents who tested the chatbot by asking questions and evaluating the answers they received. Each participant asked at least 10 questions.

Sample description:

1) Gender distribution: 7 men and 3 women
2) Age distribution:
    a. maximum age - 24 years
    b. minimum age - 19 years old
    c. average age - 21.4 years
3) Educational distribution:
    a. This year's applicants are 40%.
    b. Students - 70%.
    c. Students and graduates of NaUKMA - 60%

The total number of requests that were evaluated was 122. The following indicators were obtained as a result of the research:

1) Average rating of chatbot responses: 4 points (on a scale from 1 to 5)
2) Maximum response time: 27.8 c.
3) The average response time is 8.4 seconds.
4) The median response time is 7.7 seconds.

It is worth noting that several factors influenced the response speed, such as the stability of the network connection and the overall computer load.
After all, despite the fact that the VMM is called through the API, the local computer is responsible for using the tools. For the above reasons, the results in Section 3.6 differ from those in this section.

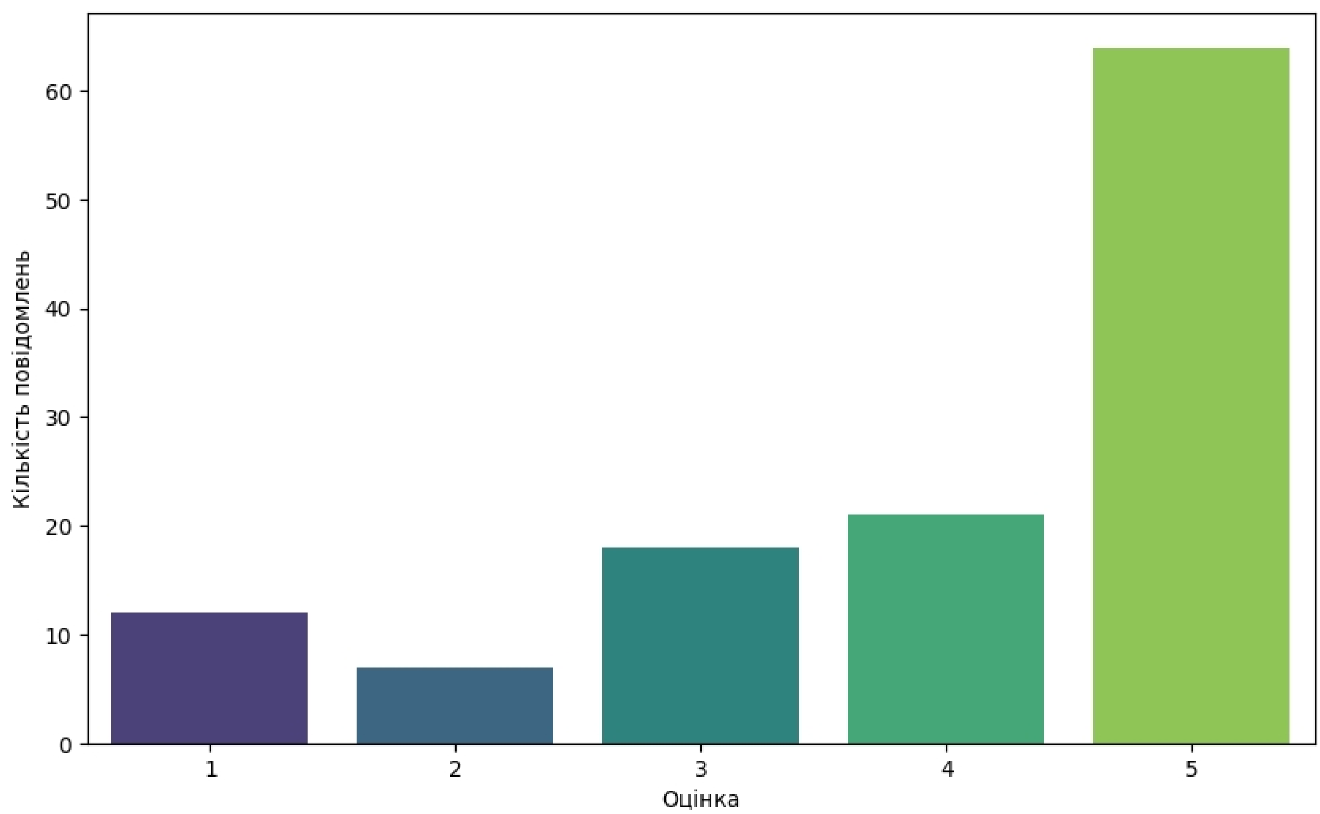

Fig. 3.8 Diagram of the distribution of chatbot response scores

Based on user feedback, it was found that the agent does not cope with clarifying questions. When the agent is asked to tell more about a certain topic, the agent makes the same call to the tool and, without receiving additional information, gives an answer without a detailed explanation.

The data obtained from this study allow us to evaluate the effectiveness of the chatbot from the point of view of different user groups, which is important for evaluating the final version and further improving the technology.

Section 4: Further work

Since agents are a relatively new technology, there is a lot of research going on to improve them. This section highlights several ways to improve the agent that were not considered in this paper but have potential.

4.1 Other investment models

There are many investment models available today [27, 28, 35]. Most of them are designed to be used either in English or in other languages. There are very few attachment models designed for use in a specific non-English language.
The Ukrainian language is no exception. There is a lack of research in this area.

In this paper, we have considered a simple version based only on the use of word attachments. Creating a full-fledged attachment model with a large model size and a large amount of training data for the Ukrainian language can improve the quality of access to information in vector databases, which in turn will improve the results of the agent tools. There is already a study of Ukrainian word attachments that describes many datasets that can be used to train a new model[36].

4.2 Adding specific tools

To improve data extraction, additional tools can also be created that contain knowledge from a specific category. In this work, information on tuition prices was extracted into a separate tool, which significantly improved the quality of answers to related questions. Therefore, if there is a discrepancy in the information from a tool that contains a large amount of information (in the context of this work, a website search tool), displaying the information in a separate tool will improve the agent's overall performance.

4.3 Use of other language models

One of the obstacles to improving the results of this version of the agent was the weak large language model. GPT-3.5-turbo has a high response speed, but it does not perform as well as older versions of language models from OpenAI and others.[37] Also, using a better language model provides a larger context window. The context window defines the maximum number of tokens in a query that the model can process. GPT-3.5-turbo has a context window of 16 thousand tokens, while GPT-4-turbo has 128 thousand, and Gemini-Pro-1.5 has 1 million [38]. Using a larger context window will help to add more information to the model to generate an answer.

4.4 Fine tuning

Fine-tuning is one of the approaches of transfer learning in which the weights of a pre-trained model are trained on new task-specific data.[39] Since training a large language model from scratch requires a very large amount of resources, it is common practice to train the model on specific data. In the context of this work, we already have ready-made data for training, namely messages from the telegrams of the "Introduction to NaUKMA" channel. Fine-tuning using this data will provide the model with new general knowledge about the university, will improve the quality of the Ukrainian language, as well as convey the style of writing used in this chat.

However, even such retraining requires a large amount of resources, so this approach was left as further work.

4.5 Reinforcement learning based on people's feedback

Reinforcement learning based on human feedback is one of the techniques for adjusting model behavior to satisfy users.[40] To use this technique, this project has already built a logic for collecting user feedback, which collects it in a separate dataset. It can be used to continuously improve the agent's performance in the future based on user feedback.

Conclusions.

This paper describes the process of creating an Admissions Assistant chatbot with a graphical interface that uses large language models and agent architecture to utilize available information and provide accurate answers. The paper covered the process of developing the initial version of the agent. Then a number of experiments were described.

Changing the response planning, where a descriptive system query was changed to a structured query with clear commands, showed improved responses and a more correct understanding of the agent's call to the tools.

Improvement of the website search tool, which consisted of several parts. First, improving the search on the vector database by changing the embedding models, comparing the results, which included the development of a new embedding model customized for use in the Ukrainian language. The results showed that the text-embedding-3-small and bge-m3 models have better results, but due to the high hardware requirements of the bge-m3 model, text-embedding-3-small was recognized as the best option for this application. Secondly, improving the methods of splitting text into smaller documents, which included testing AI21SemanticTextSplitter on a document, according to which the inclusion of additional separators improved the quality of the resulting documents, while AI21SemanticTextSplitter performed worse.

Language change, which demonstrated how the agent integrates into the use of the Ukrainian language, and which showed that the use of the Ukrainian language for all agent elements worked best.

The addition of new instruments has shown that with this approach, the information provided by the IMF is of higher quality and more structured. Accordingly, if a tool with a large amount of information and fails to provide it correctly, creating a separate tool will likely improve the agent's performance.

ReACT agents proved to be difficult to integrate with the Ukrainian language, and as a result, it was not possible to achieve stable operation of the agent using LangChain methods. It was suggested that in order to achieve this, the methods for processing model responses should be changed from English to Ukrainian.

The final version of the agent was tested using various methods and showed good results in most scenarios. However, the agent has certain limitations. First, the data the agent has access to is incomplete, so adding other data sources, as well as fine-tuning using data about NaUKMA, will expand the agent's field of expertise. Secondly, the agent has a certain query limit, so complex questions that require many tool calls can lead to exceeding the limit of the context window length. Thirdly, the model may provide incomplete answers when the documents found in the database do not contain complete information, although it is worth noting that the model indicates in such cases that the information may be incomplete.

Despite these limitations, according to the survey, the chatbot has an average score of 4 out of 5, so the current version of the chatbot can cover most of the information needs of applicants without involving human resources. At the same time, many ways to improve the agent were highlighted in Section 4. Due to the novelty of speech agent technology, new improvement methods are actively published, so this technology has great potential for further improvement.

List of references

1. Artificial Intelligence Index Report 2024 / [N. Maslej, L. Fattorini, R. Perrault et al.
2. Buchholz K. Threads Shoots Past One Million User Mark at Lightning Speed [Electronic resource] / Katharina Buchholz: https://www.statista.com/chart/29174/time-to-one-million-users/.
3. Chiara V. M. Chatbots in customer service: Their relevance and impact on service quality / V. M. Chiara, F. Poecze, C. Strauss. // Procedia Computer Science. - 2022. - No. 201. - P. 421-428.
4. Mehdi Y. Announcing Microsoft Copilot, your everyday AI companion [Electronic resource] / Yusuf Mehdi: https://blogs.microsoft.com/blog/2023/09/21/announcing-microsoft-copilot-your-everyday-ai-companion/.
5. Medical ChatBot [Electronic resource]. - 2023: https://www.johnsnowlabs.com/medical-chatbot/.
6. Conversational AI is Reshaping the Human-machine Interaction [Electronic resource] // Deloitte. 2020: https://www2.deloitte.com/cn/en/pages/innovation/articles/innovation-conversational-ai-is-reshaping-the-human-machine-interaction.html.
7. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks / [P. Lewis], E. Perez, A. Piktus, et al. Advances in Neural Information Processing Systems.
8. Augmenting large language models with chemistry tools / [A. Bran, S. Cox, O. Schilter et al.
9. Communicative Agents for Software Development [Electronic resource] / [C. Qian, X. Cong, W. Liu et al: https://arxiv.org/pdf/2307.07924.

10. Dosovitsky G. Bonding With Bot: User Feedback on a Chatbot for Social Isolation / G. Dosovitsky, E. L. Bunge. // Frontiers in digital health. - 2021. - No. 3.
11. A survey on large language model based autonomous agents / [L. Wang, C. Ma, X. Feng et al. // Frontiers of Computer Science. - 2024. - No. 18.
12. Weng L. LLM Powered Autonomous Agents [Electronic resource] / Lilian Weng // Lil'Log. - 2023: https://lilianweng.github.io/posts/2023-06-23-agent/.
13. Introduction of NaUKMA [Electronic resource] // National University of Kyiv-Mohyla Academy - Access to the resource: https://vstup.ukma.edu.ua/student-organization?so-id=8.
14. Telegram data collector v0.01 [Electronic resource] / [A. Kurochkin, A. Beck, D. Herasymuk et al: https://github.com/SanGreel/telegram-data-collection.
15. Telethon [Electronic resource] / [D. Chernov, T. Asokan, D. Pal et al: https://github.com/LonamiWebs/Telethon.
16. LangChain [Electronic resource] - Mode of access to the resource: https://python.langchain.com/v0.1/docs/get_started/introduction/.
17. Chroma [Electronic resource] - Mode of access to the resource: https://docs.trychroma.com/.
18. The Dawn After the Dark: An Empirical Study on Factuality Hallucination in Large Language Models [Electronic resource] / [J. Li, J. Chen, R. Ren et al: https://arxiv.org/abs/2401.03205.
19. Large Legal Fictions: Profiling Legal Hallucinations in Large Language Models / M. Dahl, V. Magesh, M. Suzgun, D. Ho. // Journal of Legal Analysis. - 2024.
20. Floridi L. GPT-3: Its Nature, Scope, Limits, and Consequences / L. Floridi, M. Chiriatti // Minds and Machines. - 2020. - No. 30. - P. 681-694.

21. Large Language Models: A Survey [Electronic resource] / [S. Minaee, T. Mikolov, N. Nikzad et al: https://arxiv.org/abs/2402.06196.
22. QuAC: Question Answering in Context / [E. Choi, H. He, M. Iyyer et al: Association for Computational Linguistics, 2018. - P. 2174-2184.
23. BLEU: a method for automatic evaluation of machine translation / K. Papineni, S. Roukos, T. Ward, W. Zhu: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics.
24. Lin C. ROUGE: A Package for Automatic Evaluation of Summaries / Chin-Yew Lin // Text Summarization Branches Out / Chin-Yew Lin: Association for Computational Linguistics, 2004. - P. 74-81.
25. Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study [Electronic resource] / [Y. Liu, G. Deng, Z. Xu et al: https://arxiv.org/pdf/2305.13860.
26. SneakyPrompt: Jailbreaking Text-to-image Generative Models / [Y. Yang, B. Hui, H. Yuan et al.] // 2024 IEEE Symposium on Security and Privacy (SP) / [Y. Yang, B. Hui, H. Yuan, et al. San Francisco, CA, 2024. - P. 126.
27. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation [Electronic resource] / [J. Chen, S. Xiao, P. Zhang et al: https://arxiv.org/abs/2402.03216.
28. New embedding models and API updates [Electronic resource] // OpenAI. - 2024: https://openai.com/index/new-embedding- models-and-api-updates/.
29. Learning Word Vectors for 157 Languages [Electronic resource] / [E. Grave, P. Bojanowski, P. Gupta et al.

Conference on Language Resources and Evaluation. - 2018: https://arxiv.org/abs/1802.06893.
30. Making a MIRACL: Multilingual Information Retrieval Across a Continuum of Languages [Electronic resource] / [X. Zhang, N. Thakur, O. Ogundepo et al: https://arxiv.org/abs/2210.09984.
31. AI21 Studio\'s Text Segmentation [Electronic resource] - Mode of access to the resource: https://docs.ai21.com/docs/text-segmentation-api.
32. ReAct: Synergizing Reasoning and Acting in Language Models [Electronic resource] / [S. Yao, J. Zhao, D. Yu et al: https://arxiv.org/pdf/2210.03629.
33. hwchase17/react [Electronic resource] // LangSmith. - 2023: https://smith.langchain.com/hub/hwchase17/react?organizationId=93ded755-0dce-5bd2-af47-01cef728186b.
34. Ng A. What's next for AI agentic workflows [Electronic resource] / Andrew Ng // YouTube. - 2024: https://www.youtube.com/watch?v=sal78ACtGTc&ab_channel=SequoiaCapital.
35. Text Embeddings by Weakly-Supervised Contrastive Pre-training [Electronic resource] / [L. Wang, N. Yang, X. Huang et al: https://arxiv.org/abs/2212.03533.
36. Romanyshyn N. Learning Word Embeddings for Ukrainian: A Comparative Study of FastText Hyperparameters / N. Romanyshyn, D. Chaplynskyi, K. Zakharov // Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP) / N. Romanyshyn, D. Chaplynskyi, K. Zakharov: Association for Computational Linguistics, 2023. - P. 20-31.
37. Ashrafimoghari V. Evaluating Large Language Models on the GMAT: Implications for the Future of Business Education [Electronic resource] / V.

Ashrafimoghari, N. Gürkan, J. W. Suchow // arXiv: https://arxiv.org/abs/2401.02985.

38. Gemini 1.5: Unlocking multimodal understanding across millions of context tokens [Electronic resource] / [M. Reid, N. Savinov, D. Teplyashin et al: https://arxiv.org/abs/2403.05530.
39. Quinn J. Dive into deep learning: tools for engagement / Joanne Quinn. - Thousand Oaks, CA, 2020. - 551 p.
40. Training language models to follow instructions with human feedback [Electronic resource] / [L. Ouyang, J. Wu, X. Jiang et al: https://arxiv.org/abs/2203.02155.