

Development of a Natural Language Interface for Script or API Operations Using Generative AI

Ivan Kankeu

RPTU Kaiserslautern, Department of Computer Science

***Note:** This report contains a project documentation and reflection on the portfolio task submitted for the lecture Engineering with Generative AI in WiSe 2023-24. This report is an original work and will be scrutinised for plagiarism and potential LLM use.*

1. Documentation

!!!DISCLAIMER!!!!

Please be aware that the task deadlines used in my implementation may not always be accurate. This discrepancy arises because my implementation relies on version 5.0.0 of **todocli**, which does not consistently provide deadlines in a standardized format (such as ISO 8601), but rather as durations. If the duration is in minutes or hours, the algorithm will work perfectly. Otherwise, inaccuracies can occur when attempting to add tasks or execute queries involving dates or times.

Model used: LLaMa-70B

Hint: Do not hesitate to click on the refresh button of the embedded browser if the page does not load.

1.1. Installation & Usage

Adiutor is a simple web application for managing your to-do list and performing web searches.

1.1.1. Installation

In the "adiutor" folder:

- To install dependencies: `pip install -r server/requirements.txt`
- To run the application: `flask --app server/src/app.py run --reload --port=8080`
- Open Chrome browser at URL: `http://localhost:8080`

Supported Features

The application supports the following features:

- To-do List:
 - Add a (outdoor-related) task with start, deadline, priority, context, and period
 - Merge to-do lists (contexts)
 - Move tasks between contexts
 - Prioritize tasks

- Mark tasks as done
- Remove tasks
- Show tasks
- All aforementioned actions can be performed on tasks filtered by title, context, start (after), deadline (before), task status, and position
- Web Search:
 - Summarization of the three most relevant web pages
 - Annotation of response text with sources
 - Embedded browser
- Speech Recognition and Speech Synthesis

Supported Browsers

The application has been tested on Ubuntu with Chrome (version 123.0.6312.58). It fully supports the following browsers:

- Chrome (version 2023 or later, on Windows and Linux)
- Microsoft Edge (version 2023 or later, on Windows only)

Firefox is partially supported due to the unavailability of speech recognition.

Folder Hierarchy

The "client" folder contains the front-end (web application), while the "server" folder contains the back-end. Code from the "client" folder has been compiled and added to the "server/public" folder. Hence, the "server" folder contains all the necessary code for the Python server with the web application.

- **Client/** (front-end)
 - **src/**
 - * **assets/** (CSS files)
 - * **components/** (all components belonging to the home page)
 - **calendar/** (components for to-do list)
 - **web_search/** (components for web search)
 - * **repositories/** (communication interfaces with server)
 - * **stores/** (stores to manage the application state)
 - * **views/**
- **Server/** (back-end)
 - **public/** (images, HTML, CSS, and JS files for web application)
 - **src/** (Python)
 - * **app.py** (entry point)
 - * **calendar/** (module for to-do list management)

- **weather/** (communication interface with weather API)
- **calendar.py** (transform query to operation)
- **executor.py** (operation execution)
- **todo_api.py** (communication interface with todocli)
- **prompt_template.txt** (template for multi-shot prompting)
- * **web_search/** (module for web search execution)
 - **web_search.py** (summarize search result and formulate final response)
 - **web_scraper.py** (used to perform DuckDuckGo or Google search)
 - **user_agents.txt** (user agent for web search)
- * **llm/** (communication interface with AWS Large Language Model)

1.1.2. Usage

The images below showcases various features of the application, including the calendar and web search. The settings button located at the bottom left of the page allows users to configure settings such as the API URL, API token, and system prompt, which control the output of the LLM.

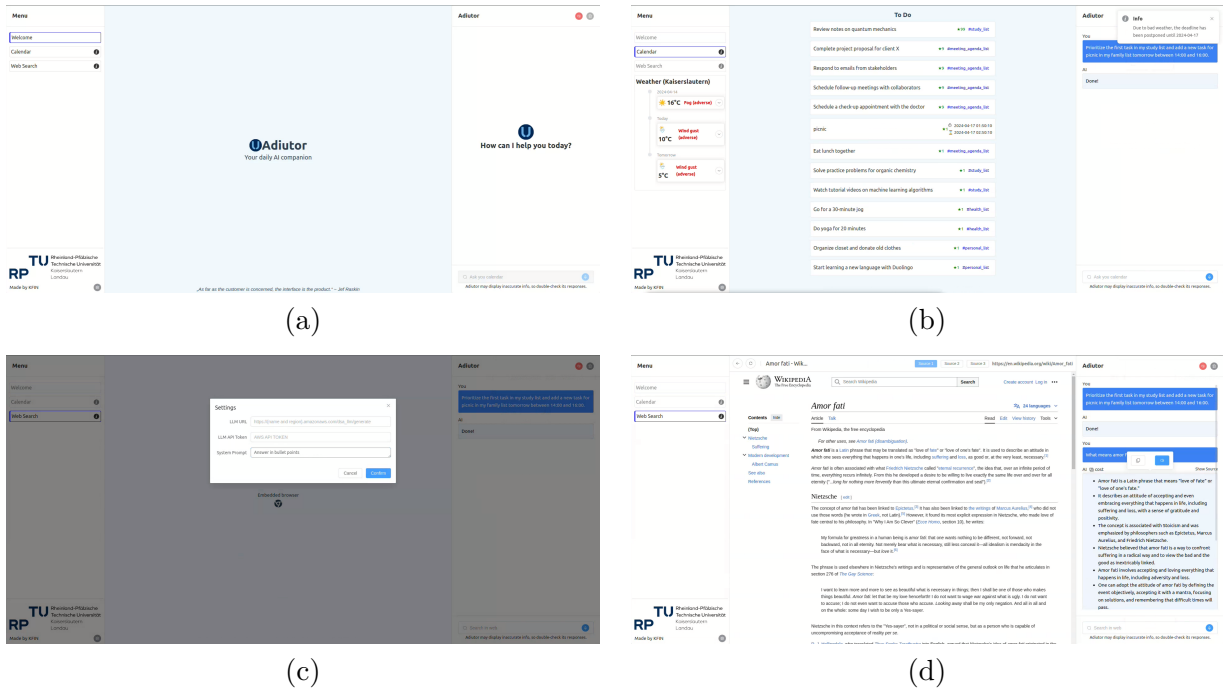


Figure 1: Preview of Adiutor

1.2. Design

The system is divided into two subsystems: the to-do list management subsystem and the web search subsystem. Each subsystem has its own pipeline to process incoming queries.

1.2.1. To-Do List Management

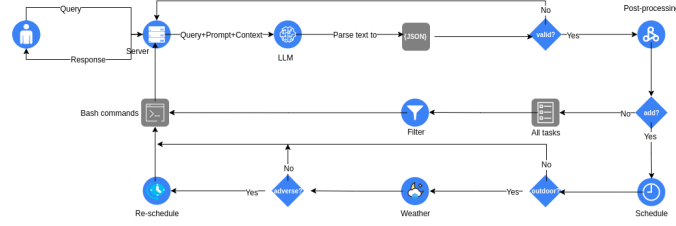


Figure 2: Workflow for to-do list management.

Upon receiving a query, the system generates a multi-shot prompt using a template, query, and context (such as the current date and task lists), and sends it to the LLM. The LLM responds with a well-structured text, which is then converted into a list of JSON objects — especially if the query has multiple instructions. These JSON objects represent valid atomic operations such as "add," "remove," "merge," "move," "prioritize," "mark," and "show." Subsequently, the JSON objects are post-processed and their values refined, for example, correcting task list names. If the operation is not an "add" operation, all tasks are retrieved as objects, filtered, updated according to the instruction, and converted into bash commands. If the task is an outdoor activity and adverse weather is expected, the day is postponed accordingly, and finally, a bash command is formulated.

When an error occurs during the process (validation or execution), an error message is generated and sent back to the client to explain what happened and what to do next. Typically, the user is inquired to rephrase their query.

1.2.2. Web Search

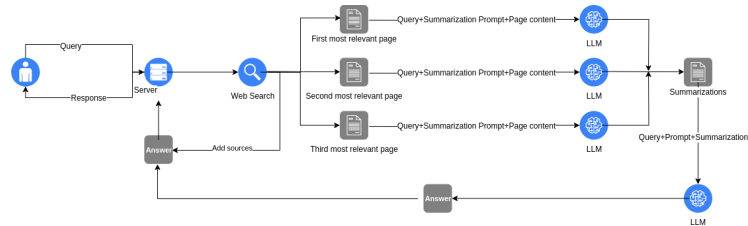


Figure 3: Workflow for Web Search.

For web search, upon receiving a new query, the system performs a DuckDuckGo or Google search, selects the three most relevant pages, summarizes them in relation to the query, and incorporates the summarizations as evidences into a one-shot prompt. This prompt is then passed to the LLM, which generates the final answer. The generated answer is annotated with sources and returned to the client along with the source pages.

1.3. Implementation

The web application is implemented using the **Vue.js** framework, while the server is implemented with **Flask**. Communication between the client and server is done through HTTP calls to the **api/query** endpoint. The data sent to the server includes the feature (calendar or web search), the query, the system prompt (used in Web Search to control LLM output), LLM URL, and API token. Conversely, the server returns a stream of data to the client at each processing step due to the long latency.

1.3.1. To-Do List Management

When a query for the calendar is received, the system converts the natural language instruction into a structured JSON format using a predefined multi-shot prompt template (see **calendar/prompt_template.txt**), which is filled with the correct (current) dates and times for each query. The LLM typically responds with a semi-structured text of the following form:



Figure 4: Transforming a natural language query into executable JSON format.

The text is parsed to JSON, validated, mapped to bash commands. An holistic analysis of system specifications indicates that users tend to formulate queries in a way that a single operation is applied to multiple elements (tasks or to-do lists). Thus, the JSON object represents an operation and attributes for filtering/creating tasks. Each task is represented by a Python object equipped with **todocli** attributes (e.g., id, title, start, deadline, context, and status) that can be easily mapped to bash commands w.r.t. operations as seen in the following image:

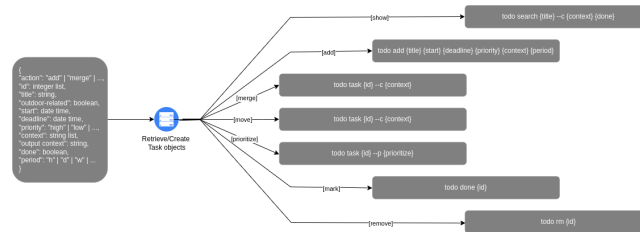


Figure 5: Converting executable JSON into bash commands.

Outdoor-related Tasks

To add an outdoor-related task (i.e., outdoor-related true) with a start time and an end time, the weather API is checked. If adverse weather conditions (such as rain, wind gusts, fog, or high temperatures) are predicted even for just one hour of the day, the start time is shifted by one day until a day without adverse weather is found or prediction forecasts are no longer available. In case of rescheduling, the client is notified.

1.3.2. Web Search

The system first performs a DuckDuckGo search using the incoming query. If this search fails (e.g., due to rate limits), a Google search is performed using a randomly selected user agent. The contents of the top three ranked pages are truncated to fit the 2048 token limit of the context window, and the truncations are then summarized using a predefined prompt template (see `web_search/web_search.py`). The resulting summaries are assigned page identifiers (of the form: "WSD-#1", "WSD-#2", etc.) and inserted into a prompt template. The page identifiers are used by the LLM for in-text annotation of sources. The final answer is then rendered to the client, which displays the LLM answer along with the sources in an embedded browser.

1.3.3. Speech Recognition & Speech Synthesis

For usability reasons, the web application leverages the Web Speech API provided by the client browser to transform speech into textual queries and server answers into speech. Thus, no substantial changes need to be made on the server side.

1.3.4. Challenges and Solutions

During the development of the system, numerous challenges were encountered, and while some were resolved with well-defined solutions, the majority required ad-hoc approaches.

Prompting

One of the first challenges was determining the most effective way to prompt the model. This proved to be challenging because the model did not react as expected compared to other models, and finding the right format for the prompt was not straightforward. After experimenting with various prompting formats, it was discovered that the LLaMa format was the most effective. In this format, the system prompt (e.g., task description or safety guidelines) had to be enclosed within «SYS» and «/SYS» tags, while examples and instructions were enclosed within [INST] and [/INST] tags. These separation tags, used during the fine-tuning of the model, proved to work effectively.

Another challenge in the calendar part was controlling the LLM's response to generate a semi-structured text that could be converted to JSON. This was achieved by lowering the entropy of the model through adjustments to the temperature (to 0.9) and using a representative set of examples. These examples helped the model relate words and concepts to each other and leverage its memorized knowledge effectively. For instance, providing the current date to the model with instructions to write the start and deadline in a specific format (%Y-%m-%d %H:%M:%S)

ensured that the model produced information in a standard format that could be easily processed by Python code.

The "Lost Middle Phenomenon" [1] was another challenge that limited the contextual information that could be passed to the model. To address this, only important and concise context information, such as the list of to-do lists and short constraint rules (e.g., predefined sets of options), were provided to give the LLM enough context to avoid extrinsic hallucinations [2]. Detailed descriptions of to-do lists and extensive information about **todocli** functionalities were not included to mitigate this issue.

Accessing External Data Sources

Another conundrum was the access of external data sources, namely those requiring an API, as many were not publicly accessible. For example, the weather API was accessed using the **wtrr.in** platform to obtain free weather forecasts for a period of three days. Additionally, predefined user agents were used for Google searches to simulate browser search queries, and special **LangChain** retrievers were employed for accessing scraper-unfriendly pages such as Wikipedia, arXiv, and PDF documents.

1.4. Evaluation & Discussion

Having successfully passed all predefined test cases, new test scenarios were designed to evaluate the system's capability to handle more complex queries.

1.4.1. To-Do List Management

As the system predominantly relies on the LLM and the post-processing of its output essentially involves merely mapping to CLI commands, the evaluation focused on the LLM's output. Five test scenarios, each comprising three queries grouped into single and multiple instructions, were devised to assess the model's ability to handle:

1. Date & Time Extraction (DTE)
2. Long Query (LG)
3. Ambiguous Query (AQ)
4. Instruction Segmentation (IS)
5. Co-reference Resolution (CR)

The evaluation results are presented in Table 2, where "valid" represents valid instructions generated by the model, and "harmless hallucination" refers to instructions with additional information inferred from the query.

| Type | Single Instruction | | | Multiple Instruction | |
|------------------------|--------------------|---------|---------|----------------------|---------|
| Test Scenario | DTE (/3) | LG (/3) | AQ (/3) | IS (/3) | CR (/3) |
| Valid | 3 | 3 | 2 | 2 | 3 |
| Harmless Hallucination | 1 | 0 | 0 | 0 | 0 |

Table 1: Evaluation results

Overall, the model demonstrated a tendency towards "harmless hallucinations" and displayed relatively good robustness towards long queries. It effectively segmented queries into multiple independent instructions and demonstrated good co-reference resolution. However, it occasionally produced instructions that did not adhere to the predefined format, particularly for ambiguous queries, and sometimes relied too heavily on the provided multi-shot examples, leading to inadequate generalization. Further details of the evaluation results can be found in Appendix.

1.4.2. Web Search

Due to the high cost (4 LLM API calls) associated with this feature, the evaluation focused only on the system’s ability to resolve factoid and non-factoid queries. For factoid queries, the system successfully retrieved the current temperature in a city but presented the answer in a less concise format. Non-factoid queries involved retrieving and summarizing information on a subject, which was done correctly. However, the model failed to annotate the text with sources. Despite the system’s ability to synthesize information being relatively good, the resources used were disproportionate. The complete evaluation report is provided in Appendix.

2. Reflection

2.1. Introduction

Over the past seven weeks, I have dedicated my efforts to developing a natural language interface for executing operations, as part of my portfolio project. The project focused on two main objectives: generating bash commands for a to-do list manager and facilitating web searches. The primary aim was to bridge the gap between complex API interfaces and human language, leveraging the natural language understanding capabilities of LLM (Large Language Model). Despite initially planning a six-week timeline for the project, I encountered unexpected challenges that required significant code revisions and an extension of the project timeline. However, I successfully met the deadline and achieved my objectives to a satisfactory extent. This document serves as a reflection on my work and provides a critical assessment of the results attained.

2.2. The Most Interesting Thing I Learned

Throughout the project, I acquired various insights, but if I were to pinpoint one, it would undoubtedly be prompt engineering. While I had previous experience with prompting LLMs, working with a small and sensitive model posed unique challenges, particularly when structured output was required. To guide the model towards generating text that adhered to a predefined

machine-readable format, I had to control and guide its entropy and attention. My goal was to establish connections between the problem at hand and the model's accumulated knowledge during training, mitigating "bad hallucinations." I found that using frequently occurring terms aided the model in better understanding the task. Additionally, I observed that smaller models, like the one used in LLaMa, were more susceptible to repetitions and overfitting in few-shot or multi-shot settings, emphasizing the importance of having lexically and semantically diverse examples in the prompt.

Among the many experiences during the project, the most surprising was the significant performance gap between the provided model and state-of-the-art models (e.g., GPT) in language understanding. Simple prompts that worked seamlessly with GPT were misinterpreted by LLaMa, highlighting the ambiguity in prompt formulation and LLaMa's lack of background knowledge. This substantial performance gap prompted me to reassess my approaches to task-solving, since parts of the portfolio that I assumed to be straightforward turned out to be the most challenging and time-consuming. The ambiguity issue stemmed not only from incorrect formulation but also from the inherent nature of human language, which is not conducive to conveying formal, well-defined, and contextualized information.

2.3. The Part of the Portfolio I am Proud of

Although every aspect of the portfolio presented its challenges, the implementation of the web search functionality stands out as a source of pride for me. It is not because it represents the pinnacle of technical achievement, as there is always room for improvement, but rather because it addresses the shortcomings of state-of-the-art web search systems I had used previously (such as `copilot.microsoft.com` and `perplexity.ai`). These systems lack an embedded browser, making it cumbersome and time-consuming to verify sources and ensure the correctness of results provided by LLM. Despite running on the most advanced GPT models, they are often subject to intrinsic hallucinations [2] or overlook critical details during summarization. Therefore, providing users with the ability to review sources with a single click greatly enhances usability. Additionally, users can control the format of the answer without altering the query itself, a feat impossible on other platforms. An illustrative example is when a user seeks to retrieve page contents and instructs the model to generate questions based on them. In this scenario, the actual query information could be buried in the task description and hamper the web search. Hence, the query needed to retrieve the page contents should be separated from the model instruction.

Surprisingly, the aspect of the portfolio I am most proud of was not the most challenging to implement. To be candid, I encountered more difficulties with the calendar task, facing several challenges:

1. Generating commands effectively using the model was a significant challenge. I experimented with various strategies, including providing descriptions and examples of **todocli** commands and options. While this approach worked well for atomic commands, it proved inadequate for complex instructions like move operations constrained by filters. To address this, I defined high-level commands that aggregated smaller ones and post-processed the model's output to map high-level commands to atomic ones which significantly improves accuracy.

2. Another obstacle encountered was minimizing the number of prompts to faster response times and reduce costs. To this end, the sequence of prompts that was executed to step-wise go through the entire **todocli** description was summarize into one single oversimplify prompt. The prompt through examples is leading the model to learn in-context the command types and options available that could be used to generate a valid command. This enhances processing time and improves the user experience.
3. Another challenge was the implementation of the communication interface between the server and browser client. To keep the user experience as fluid as possible and reduce the perceived latency, I used HTTP Streaming instead of classical REST to send the data chunk by chunk as soon as they were ready on the server. Functions on the client and server sides were implemented to send, parse, and process JSON data sent through streaming. The format of JSON has also been adapted to align with the method of communication. This helped to send server feedback (like rescheduling of an outdoor activity due to weather conditions) to the client during the processing of a request.

2.4. Revisions to Design and Implementation

The project's development was far from a smooth and meticulously planned journey; rather, it was fraught with obstacles and unexpected twists. Initially, I embarked on implementing the web application and laying down basic server functionalities without a clear understanding of how the model would behave. Prior to the project's implementation, I tested various prompts on GPT to ensure task feasibility. However, I was surprised to find that prompts effective on GPT were not well understood by LLaMa. Consequently, I had to overhaul my implementation significantly to mitigate reliance on the model, ensuring robustness in command generation despite model weaknesses and instabilities. As a result, the model was used solely for natural language understanding and generating intermediate representations, which were then processed using conventional Python code. Towards the project's conclusion, I introduced support for queries containing multiple instructions, requiring adjustments to the prompt and command generator functions with minimal impact on the codebase.

In spite of the adjustments I have made, there are still some points that I would have liked to handle differently. One such point is the use of the LLM in the command generation process. Specifically, I would like to address instability in the model output by using the Tree of Thoughts (ToT) in a multi-agent setting. In this setup, agents could be used to break queries into atomic instructions and verify whether the extracted data (such as to-do list names, dates, etc.) are correct and complete. Since I discovered the generative AI field, I have been fascinated by the idea of replacing or supporting researchers in their daily work using an LLM in a RAG setting for targeted web searches. This could prove to be a valuable area for future improvement. my current implementation of web search is a basic solution that only supports simple query-based searches. An enhancement could involve the ability to define, instead of a query, an information retrieval model specifying which types of information need to be retrieved and how it should be done. This specification model would then be executed by an engine that orchestrates the web

search on predefined web pages, extracts useful data through structure-aware or semantic chunking, and feeds it to an LLM-based multi-agent system to retrieve information in a structured format (e.g., JSON or XML).

2.5. Ethical Considerations in Code Generation

Within the realm of generative AI, code generation stands out as one of the most promising yet challenging applications. Its potential lies in the significant advancements it promises to bring to the software development industry, particularly in terms of productivity and cost reduction. However, the deployment of this technology is hindered not only by technological challenges but also by profound ethical concerns.

One of the primary ethical dilemmas raised by code generation is the blurring of the lines between human and machine-created content. This raises complex questions regarding copyright and intellectual property rights, which are already challenging to address from both legal and technical perspectives. Most code generation models today are large, intricate systems with billions of parameters, lacking transparency and interpretability. This opacity makes it difficult to understand the reasoning behind the model's output and trace its source.

The lack of transparency rubs off on the liability issue by making it difficult to identify the mistake causing the fault, as there is no rational highlighting of the generation process provided by the model. In the event of security vulnerabilities or other issues caused by generated code, it becomes unclear who should be held accountable. Is it the user of the model, the engineer who designed and trained it, or the creator of the training data set?

Texts generated by code generators exhibit, like other generative models, four types of biases [3]: cultural bias, linguistic bias, temporal bias, and political bias. These biases, which are amplifications of biases existing in training data [4], may perpetuate or reinforce social inequalities and cause significant harm to humans and society when deployed on a large scale. Therefore, following principles of Explainable and Responsible AI throughout the process of the data set creation, model training and code generation (in production) is essential to ensure that the outcomes align with current societal values and norms.

2.6. The Most Exciting Topic and Outlook

The topics addressed in the course beyond enriching my technical knowledge about generative models have also enhanced my critical sense on their development and usage. Among the topics, the one that excited me the most was parameter-efficient fine-tuning. Because it provided insights into how the model architecture influences learning and how each layer sequentially extracts abstract representations fed to the next layer. Another intriguing discovery was the concept of intrinsic dimensions in language models, which, when viewed mathematically, represent a form of hidden shared structure that we try to approach with computationally efficient model architectures.

If there is one thing I would like to learn more about, it is LLM agents. Many real-world problems are complex and involve multiple dependent and independent steps, often requiring collaboration among different agents. Therefore, it would be interesting to gain an overview of the state-of-the-art techniques and frameworks currently used to build agent-based systems.

3. Multiple Choice Question

1. What is the effect of longer pre-training time on language models?
 - a) It increases the fine-tuning time needed to achieve d90 performance.
 - b) It decreases the intrinsic dimensionality.
 - c) It makes fine-tuning on downstream tasks more difficult.
2. Which of the following statements is true?
 - a) Constitutional AI incorporates human feedback during reinforcement learning.
 - b) The policy model used in RHLF is the pre-trained model.
 - c) The KL divergence shift penalty in PPO helps in preventing catastrophic forgetting.
3. Which concept about prompts is true?
 - a) Adding context to the prompt lowers the entropy.
 - b) Jailbreaking is a technique used against prompt hacking.
 - c) During prompt tuning, the model parameters are trained.

References

- [1] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023.
- [2] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, March 2023.
- [3] Emilio Ferrara. Should chatgpt be biased? challenges and risks of bias in large language models. *First Monday*, November 2023.
- [4] Roberto Navigli, Simone Conia, and Björn Ross. Biases in large language models: Origins, inventory, and discussion. *J. Data and Information Quality*, 15(2), jun 2023.

A. Appendix

A.1. Evaluation Results of To-Do List Management

| Query | Ground Truth | Prediction |
|---|--|--|
| Date & Time Extraction | | |
| What sport activities do I have to do next week? | show -context sport -start 2024-04-15 00:00:00 -deadline 2024-04-21 23:59:00 | show -context sport -start 2024-04-15 00:00:00 -deadline 2024-04-21 23:59:00 |
| Add a daily task for exam revisions between 21:00 and 23:00 | add -title exam revisions -outdoor-related false -start 2024-04-14 21:00:00 -deadline 2024-04-14 23:00:00 -period 1d | add -title exam revisions -outdoor-related false -start 2024-04-14 21:00:00 -deadline 2024-04-14 23:00:00 -period 1d |
| Move tasks starting in 2 hours to priorities | move -start 2024-04-14 21:00:00 -context priorities | move -start 2024-04-14 21:00:00 -context priorities |
| Long Query | | |
| Today I am very busy so remove all tasks in my sport list | remove -context sport | remove -context sport |
| Because the weather now is horrible set the priority of sport activities to lowest. | prioritize -context sport -priority lowest | prioritize -context sport -priority lowest |
| We didn't eat anything yesterday. Add a task for cooking today between 10:00 and 12:00. I will invite some friends. | add -title cooking -outdoor-related false -start 2024-04-14 10:00:00 -deadline 2024-04-14 12:00:00 | add -title cooking -outdoor-related false -start 2024-04-14 10:00:00 -deadline 2024-04-14 12:00:00 -context social |
| Ambiguous Query | | |
| Move no no no. Please mark all tasks in my work list as done | mark -context work -done true | mark -context work -done true |
| Add task to sport list by merging cooking with sport list | merge -context cooking -output context sport | merge -context cooking -output context sport |
| Prioritize the first task by moving it to priorities | move -id [1] -output context priorities | move -title [1] -output context priorities |
| Instruction Segmentation | | |
| Empty my sport list. | remove -context sport | remove -context sport |
| Add a task for pizza delivery. Show all tasks. | \nadd -title pizza delivery -outdoor-related false\nshow | \nadd -title pizza delivery -outdoor-related false\nshow |
| Prioritize and mark as done the first task. | prioritize -id 1\nmark -id 1 -done true | prioritize -id 1\nmark -id 1 -done true |
| Before removing the first task show me all tasks. | show -context all\nremove -id [1] | show -context all |
| Co-Reference Resolution | | |
| Show the tasks in cooking list and prioritize the first one of them. | show -context cooking\nprioritize -id [1] -priority highest -context cooking | show -context cooking\nprioritize -id [1] -priority highest -context cooking |
| I have a cooking list merge it with my sport list. | merge -context cooking -output context sport | merge -context cooking -output context sport |
| There are a cooking list and a sport list prioritize the second task in the latter. | prioritize -id [2] -priority maximum -context sport | prioritize -id [2] -priority maximum -context sport |

Table 2: Evaluation results

A.2. Evaluation Results of Web Search

A.2.1. Factoid Query

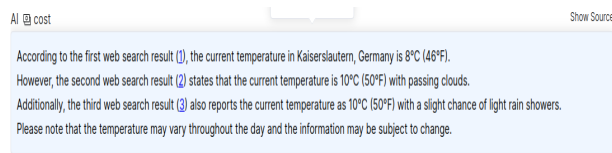




Figure 6: Query: What is the current temperature in Kaiserslautern?

A.2.2. Non-Factoid Query

AI  cost Show Source

According to the web page contents, quantum computers are not yet production-ready (2). While there have been advancements in quantum computing technology, current machines are still prone to errors and are in the "noisy intermediate-scale quantum" phase of development (1). The holy grail is a large-scale quantum computer that can correct its own errors, but this has not yet been achieved (3). However, researchers are actively working to overcome the challenges that stand in the way of quantum computing's widespread adoption, and some experts believe that general-purpose quantum computers may be available earlier than anticipated (4).

Figure 7: **Query:** Are quantum computers production-ready?

AI  cost Show Source

| Year Life Expectancy (years) | | |
|------------------------------|-------|--|
| 1950 | 66.69 | |
| 1980 | 75.42 | |
| 2000 | 79.58 | |
| 2010 | 81.1 | |
| 2020 | 82.04 | |

(Note: The data is based on the reference year 2020 and is provided by the United Nations.)
Additionally, here are some key points about life expectancy in Germany:

| Aspect | Value |
|--|--|
| Life expectancy in 2020 | 82.04 years |
| Growth rate from 2022 to 2023 | 0.19% |
| Highest life expectancy recorded | 81.57 years (in 2020) |
| Lowest life expectancy recorded | 66.69 years (in 1950) |
| Regional differences | New federal states: 83.5 years (women), 79.4 years (men) Old federal states: 82.5 years (women), 77.4 years (men) |
| Social differences | Highest income group: 85.2 years (women), 81.5 years (men) Lowest income group: 80.3 years (women), 76.8 years (men) |
| Comparison to other European countries | Germany is 2.7 years below Switzerland (highest life expectancy) |

Please note that the information provided is based on the given text and does not cover other aspects of the topic. If you have any further questions, I suggest consulting additional sources or using the reference provided at the end of the text.

Figure 8: **System Prompt:** Answer with summary tables. **Query:**What is the evolution of life expectancy in Germany over years?

Informed Consent of Participation

You are invited to participate in the field study **LLM Education** initiated and conducted by Applied Machine Learning group. The research is supervised by **Sebastian J Vollmer**. Please note:

- Your participation is entirely voluntary and can be withdrawn at any time.
- The field study will last approximately 6 weeks.
- We will record the documentation submitted by the student at the end of the portfolio exam.
- All records and data will be subject to standard data use policies.

If you have any questions or complaints about the whole informed consent process please contact Sebastian J Vollmer (E-Mail: sebastian.vollmer@dfki.de).

Purpose and Goal of this Research

Ability to create multiple choice questions of LLMs. Can we prompt an LLM to create good multiple choice questions? Your participation will help us achieve this goal. The results of this research may be presented at scientific or professional meetings or published in scientific proceedings and journals.

Participation and Compensation

Your participation in this study is completely voluntary and is unpaid.

Procedure

After confirming the informed consent the procedure is as follows:

1. Student submits the exam with deliverables
2. Student also creates questions of their choice based on instructions of good multiple choice questions.
3. We compare LLM's output to MCQs created by students.

The complete procedure of this field study will last approximately 6 weeks.

Risks and Benefits

There are no risks associated with this field study. We hope that the information obtained from your participation may help to bring forward the research in this field. The confirmation of participation in this study can be obtained directly from the researchers.

Data Protection and Confidentiality

We are planning to publish our results from this and other sessions in scientific articles or other media. These publications will neither include your name nor cannot be associated with your identity. Any demographic information will be published anonymized and in aggregated form. Contact details (such as e-mails) can be used to track potential infection chains or to send you further details about the research. Your contact details will not be passed on to other third parties. Any data or information obtained in this field study will be treated confidentially, will be saved encrypted, and cannot be viewed by anyone outside this research project unless we have you sign a separate permission form allowing us to use them. All data you provide in this field study will be subject of the General Data Protection Regulation (GDPR) of the European Union (EU) and treated in compliance with the GDPR. Faculty and administrators from the campus will not have access to raw data or transcripts. This precaution will prevent your individual comments from having any negative repercussions. During the study, we log experimental data, and take notes during the field study. Raw data and material will be retained securely and compliance with the GDPR, for no longer than necessary or if you contact the researchers to destroy or delete them immediately. As with any publication or online-related activity, the risk of a breach of confidentiality or anonymity is always possible. According to the GDPR, the researchers will inform the participant if a breach of confidential data was detected.

Identification of Investigators

If you have any questions or concerns about the research, please feel free to contact:

Sebastian J Vollmer

Principal Investigator Trippstadter Str. 122

67663 Kaiserslautern, Germany sebastian.vollmer@dfki.de

☐ I understand the explanation provided to me. I understand and will follow the hygiene rules of the institution. I understand that this declaration of consent is revocable at any time. I have been given a copy of this form. I have had all my questions answered to my satisfaction, and I voluntarily agree to participate in this field study.

☐ I agree that the researchers will and take notes during the field study. I understand that all data will be treated confidentially and in compliance with the GDPR. I understand that the material will be anonymized and cannot be associated with my name. I understand that full anonymity cannot be guaranteed and a breach of confidentiality is always possible. From the consent of publication, I cannot derive any rights (such as any explicit acknowledgment, financial benefit, or co-authorship). I understand that the material can be published worldwide and may be the subject of a press release linked to social media or other promotional activities. Before publication, I can revoke my consent at any time. Once the material has been committed to publication it will not be possible to revoke the consent.

Signature

Place, Date