LLM Practical: Report Phase 1

Frederik Schittny, Thomas Marwitz

14 May 2025

1 Introduction

The developed system will assist users in navigating the KIT campus by offering a campus map with a natural language interface. This will significantly improve the usability of the current campus map, which is only able to show the location of certain buildings on a map, but can not be used to determine a building's address, use, or get directions to navigate to a building. The data that is used as the basis of the new system consists of data scraped from the current campus map, enriched with additional data pulled from OpenStreetMap by reverse geocoding with Nominatim. This report concludes the first phase of the project and outlines the tasks and data processing that the system will implement, as well as the creation of an evaluation dataset and the testing methodology.

2 Task Specification

The system will be able to perform several different tasks. Although all of these tasks will be based on the retrieval of information from the assembled database, the system's capabilities will not be limited to simple question answering.

2.1 Retrieve Static Information

Static information retrieval will be based on retrieval augmented generation (RAG) and focus on conveniently serving static information to the user from the database, based on natural language input and output. Some example prompts include:

- Where is building 50.34?
- What is the address of InformatiKOM 1?
- When is the Mensa open?
- Is the Egon-Eiermann-Hörsaal accessible with a wheelchair?

2.2 Use of Contextual Information

While the information collected in the database is static, it is highly advantageous for the system to have access to current and contextual information about the user. This can make interaction with the system more natural and intuitive. Since some of this information is difficult to collect and process, several use cases have been assigned a low priority (compare section 2.4.4 and section 2.4.1).

2.2.1 Current Time

One of the most basic contextual information to provide is the current day of the week and time of day. This allows the system to answer prompts about opening hours like

- Is the Mensa open right now?
- How long until the library closes?

with more natural and direct answers rather than simply returning the fixed opening hours.

2.3 Tool Usage

Based on the information retrieved with RAG, the system will be able to use (and possibly combine) different tools that increase user convenience.

2.3.1 Create Link for Navigation

As one of the most important user goals is navigating to specified buildings, the system will be able to generate clickable navigation links. These links directly start a route to the described destination in a (potentially selectable) navigation app (e.g. Google Maps, Apple Maps, OpenStreetMap). Prompts to trigger this tool usage might look like this:

- I want to go to the central library.
- How do I get to the Carl-Benz-Hörsaal?

2.3.2 Navigate to Website

With this tool, the system can take the user directly to the website associated with a building or institution, based on a prompt like

- Open the website of the Forschungszentrum Umwelt.

This feature is dependent on the technical ability to proactively open a website on the user's device. If this capability is not given due to the system's design and setup, this feature would simply return a clickable URL, making it identical to the functionality described in section 2.1.

2.4 Additional Capabilities

While all features described above are considered essential for the system's usability and will be developed first, many additional useful features are imaginable. These are described below and are assigned medium or low priority. This is either because they do not provide essential functionality or because their implementation is expected to be especially challenging.

2.4.1 Warn About Closed Buildings (medium priority)

When the system has access to the user's current time as described in section 2.2.1, it can inform the user that a building is closed when they prompt the system for information about the building's opening hours. More importantly, the system could issue a warning when the user asks for a navigation link to a currently closed building, preventing users from showing up at a closed destination.

2.4.2 Building amenities and affiliation (medium priority)

Currently, the database only contains some information about building amenities and facilities. If more information like this can be collected and added to the dataset, it would allow the system to answer prompts about the availability of restrooms, kitchens, etc. This would expand the system's use cases from purely informative to more exploratory scenarios. As part of this, the system could also answer questions about groups of buildings, like finding all buildings associated with a certain discipline or institute.

2.4.3 Contact Information (low priority)

The dataset currently contains very little usable contact information associated with buildings. However, for many buildings and institutes, contact information like phone numbers and email addresses exists. If these are introduced into the dataset, it would allow the system to return a clickable link that directly puts a user in contact with a person responsible for a building. This could also be the system's reaction to prompts that it can not answer itself, but that it suspects the user can get answers for, by contacting the responsible person.

2.4.4 Current Location (low priority)

Access to the user's current location would allow the system to directly answer questions about distance or routes to locations instead of relying on external services as described in section 2.3.1. This would, however, involve the use of complicated external tools, as the LLM itself is likely incapable of producing reliable navigation based on geocoordinates alone.

2.4.5 Accessing external information (low priority)

The websites that are part of the dataset, as well as the usage of external tools like search engines, could allow the system to collect additional information on institutes or individual buildings and provide this information to the user depending on the context. It is, however, unclear how much additional information about the buildings themselves, their function, and locations can be acquired this way. Most information scraped from web searches and websites would likely fall outside of the system's scope.

3 Input and Output Processing

Besides the system's components implementing the functionalities described in section 2, the system will contain several components dealing with pre- and post-processing of the natural language inputs and outputs.

3.1 ASR Input

The input of the LLM will be strictly text-based, and no multi-modal approach will be pursued. To allow the system to process spoken prompts, an automatic speech recognition (ASR) will transcribe audio inputs to text. Depending on transcription accuracy and response latency, different ASR implementations (e.g. server-based or local) will be considered to ensure a high response quality and low answer times for the complete system.

3.2 TTS Output

In order to allow for spoken outputs, the system can use a text-to-speech (TTS) engine to convert the LLM's text output into spoken language. This processing step is prioritized lower than the ASR input, since some functionality, like the generation of navigation links or the opening of websites, does not work in a hands-free operation anyway.

3.3 Multilingual In/Output

Because many of the buildings' and streets' proper names are in German, the system's internal language is also chosen to be German. It is expected that this will yield higher performance. This assumption could be evaluated qualitatively by switching the internal language to English and comparing the resulting performance. To support input and output in other languages, a translation component can be added after the ASR input processing and before the TTS output. Since this is a less essential use case, the implementation of this processing step is prioritized lower than the ASR and TTS processing.

4 Test Set

The overall test set consists of pairs of system prompts and exemplary answers. Both prompts and answers are created directly from the dataset by automatically filling fixed template sentences with a Python script. To diversify the prompts' wording and formulation, the filled template sentences are given to an LLM with the task of correcting any grammatical imperfections caused by the template filling and rephrasing the prompts while keeping the contained information.

This procedure allows for creating a large evaluation dataset with multiple test prompts per data sample while only taking minimal manual effort. It also makes it easy to produce more test prompts

when the dataset is expanded. The large amount of prompt-response pairs also opens up the opportunity to use part of the dataset as training data in a later development phase. This training could include fine-tuning of the LLM or fixed-LM prompt tuning with discrete or continuous prompts.

While the written prompts can already be used to evaluate the core components, this would not consider any errors potentially introduced by the ASR and TTS components. To test the system end-to-end, a selection of prompts is recorded by human speakers to be fed into the ASR component at the beginning of the processing pipeline.

The test set is divided into three parts to evaluate prompts of different levels of complexity and characterize different aspects and components of the system.

4.1 Single-Turn

Single-turn prompts consist of a single prompt-response pair where all necessary information needed for the system to generate a proper response is directly included in the user's prompt or provided as contextual information. The prompts in the test data cover all core functionality described in section 2. Testing individual features can mainly unveil which types of tasks the system is struggling with, while also simulating very short and sporadic user interaction.

4.2 Multi-Turn

The multi-turn dataset consists of a series of prompt-response answers, where all but the first prompt require information from the conversation context to generate a correct response. The information left to the context in these follow-up prompts is typically either the building identifier (e.g. "And when does it open?") or the kind of task that is supposed to be performed (e.g. "And how about building 40.21?"). These multi-turn tests simulate more advanced user behavior and are more challenging for the system as they require referring to context in addition to retrieving and processing the correct information from the dataset.

4.3 Misinformation Robustness

Some of the evaluation will focus on the robustness of the system against providing misleading or wrong information to the user. Because the potential impact of misinformation is very low, this evaluation will be kept qualitative and exploratory with no pre-defined test set. Instead, challenging prompts will be made up and documented "on the fly". The only challenging prompts in this regard that are already included in the single- and multi-turn tests are asking for plausible but non-existent information, with the expectation that the system can decline answering these prompts with a generic answer instead of hallucinating a response.

Other scenarios that might be included in the misinformation robustness tests include providing information to the system that conflicts with the dataset's ground truth, asking for non-supported functionality, pointing out (non-existent) errors in the system's responses, and distracting the system by burying the relevant information and instructions in superfluous information noise.

5 Evaluation

The test set described in section 4 will be used to evaluate the system's performance in the form of the metric described below. This gives an estimate of how well the final system will work in actual user interaction and makes it possible to compare different implementations to improve the system over time.

5.1 Metrics

Since the system never answers with a sequence of predefined tokens but always outputs a natural language text, the evaluation will be based on individual answer scoring rather than predefined scores like the F-measure. The only exception from this is the RAG component, which outputs a

list of data samples that were identified as relevant and that can be evaluated using a quantitative rating like the F-measure.

The simplest evaluation metric for the complete system is a simple binary success score given for each prompt or series of prompts. It is not useful to directly compare the system's response to the example response included in each test sample as described in section 4, as the system will likely not be fine-tuned and will therefore produce a wide variety of response phrasings. One simple option is to assign the effectiveness score by manually comparing the system's response with the information given in the exemplary response associated with the provided prompt.

We will also explore the option of using an LLM as a judge to score the test answers based on the provided reference answer. One potential candidate is the Pydantic evaluation framework. This evaluation will also be a simple qualitative success rating, giving the answers a binary pass/fail score.

5.2 Analysis

To make the system's evaluation actionable and improve the system in meaningful ways, a categorization of errors made during testing might be helpful. An error categorization is not part of evaluating the system's performance and will therefore likely only be conducted anecdotally or for certain use-cases. Potential error categories include:

- task identified incorrectly / wrong tool used
- incorrect information / hallucination
- bad response time / timeout