Human-Machine Dialogue: Project Report

Roman Simone (247181)

University of Trento
Email: simone.roman@studenti.unitn.it

I. Introduction

This report presents LLamaFit, a human-machine dialogue system designed to help users plan and manage their workout programs.

The core idea behind LLamaFit is to help people by providing personalized schedule for training and exercising, guiding as a human personal trainer. Typically, people seek personal trainers for two main reasons: to get a comprehensive workout plan covering multiple sessions or to find specific exercises targeting certain muscle groups. LLamaFit addresses both by generating customized training schedules and suggesting exercises suited to individual preferences. Furthermore LLamaFit can handle a list of favorite exercises of the user.

The classic exchange types for LLamaFit are: "Can you give me a full workout plan? I usually train 2 times a week", or, "Can you give me an exercise to train the chest?". The system searches on its database for the most suitable exercises. To enhance personalization, the system also allows users to save and retrieve their favorite exercises. A user could ask, "Show me my favorite exercises for glutes" and LLamaFit will return a list of previously saved exercises along with instructions.

This system is designed for anyone who wishes to improve their physical fitness, no matter their experience level. With personalized guidance and easy access to expert-backed workouts, it acts as a virtual personal trainer that's always available, anytime and anywhere, helping people to reach their fitness goals.

II. Conversation Design

A. Dialogues

LLamaFit has been designed to handle three main types of dialogues:

- **Full workout plan**: The user can request a complete weekly training program by specifying the number of sessions and their fitness level. The system then provides a structured workout plan, including various types of exercises such as strength training, cardio, and stretching.
- **Get specific exercise**: The user can ask for a particular exercise based on specific criteria, such as the muscle group it targets, their fitness level, and the available equipment. The system retrieves the most suitable exercise from the database, ensuring it aligns with the user's preferences. The system provides the exercises from the database that best match with the user info. The dialogue could be continued on user initiative or by the system with the purpose of giving more information about the exercise or to add the exercise to user favorite list.
- **Handle favorite list exercise**: The user can request to view his favorite exercises based on specific categories, such as strength training or stretching, as well as by targeted muscle groups. Additionally, the user can add or remove exercises from thir list. If the user wants to add an exercise that does not exist in the database, the system will propose to save the new exercise before adding it to their favorites.

B. Interaction proprieties

To ensure a smooth and user-friendly conversational experience, it is essential to manage key interaction properties effectively.

1) Initiatives

The dialogue with LLamaFit follows a mixed-initiative approach. The system is designed not only to respond to user-initiated queries but to proactively suggest actions as well, such as adding an exercise to their favorites or asking if the user wants more details about a particular exercise.

2) Engagement

The system is designed to enhance user engagement by responding in a polite and friendly manner. To further improve interaction, it proactively suggests relevant actions and ensures clarity in every response, making conversations more interactive and user-friendly. Additionally, LLamaFit can request

user feedback, allowing it to assess and refine the quality of interactions over time.

3) Acknowledgement and confirmation

The system provides acknowledgment once an intent is confirmed. When the user supplies all the necessary information to complete a request, the system summarizes the details to ensure understanding. Before delivering the final response, it offers a brief recap of the input information, ensuring accuracy and clarity.

4) Fall-back Responses

The system is designed to handle out-of-context user inputs effectively. If the user provides input that is entirely outside the system's scope, it responds by informing them that the request is not understood while reminding them of its capabilities. If the input contains both relevant and irrelevant information, the system keeps the conversation in the relevant intent and ignores the irrelevant information.

5) User profile

LLamaFit is designed to handle both under-informative and over-informative users. When interacting with an under-informative user, the system proactively asks for any missing details before providing a final response. For over-informative users, the system efficiently processes multiple intents by organizing the information into a structured list. Then it is able to create a compact and precise response that unifies every request of the user.

6) Conversational marker

To create a more natural, human-like dialogue, the system incorporates conversational markers in its responses. For example, it may begin with phrases like "Well! ..." or "Great! ..." to make interactions feel more engaging and conversational.

7) Error Recovery

When the system fails to retrieve a matching response from the database, it proactively suggests a solution to the user. In the case of an unsuccessful exercise search, it summarizes the provided input, informs the user that no matching exercise was found, and asks whether the details are correct or need to be adjusted. Similarly, if an attempt to add an exercise to the favorites fails because the exercise does not exist in the database, the system reviews the input data and offers the user the option to add a new exercise instead.

III. Conversation Model

The conversational pipeline is structured into three key components (see Fig.1):

- **NLU** (*Natual Language Understanding*): Responsible for processing user input by performing intent classification and slot filling.
- **DM** (*Dialogue Manager)*: Handles intents and slots received from the NLU, tracks the conversation state using a state tracker, selects the next best action, and, when necessary, retrieves and filters relevant data from the database before responding.
- **NLG** (*Natural Language Generato*): Generates user responses by taking the next best action from the Dialogue Manager and incorporating any relevant data when necessary.

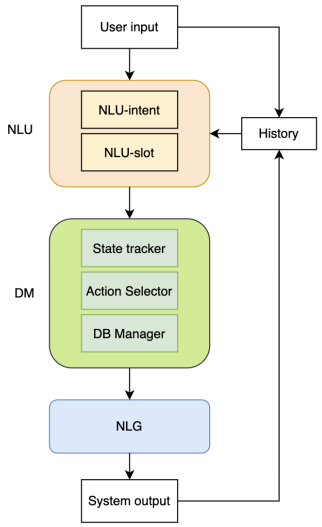

Fig. 1: System architecture

All components are implemented using the LLama3 [1] LLM model, running locally with Ollama [2]. I chose to use LLama locally as it allows for more flexibility and repeated testing without limitations.

For each component, I created different based on intents. In total i choose 8 intents (see Table II). Each prompt follow this template:

- *Context of the Component*: Defines the role and function of the component within the conversational pipeline.
- *Generation Rules* : Establishes guidelines for how the component should process and respond to input.

- *Additional Information*: Provides further details relevant to the component's behavior and functionality.
- *Examples*: Includes sample inputs and outputs to guide the model's responses.

A. NLU

The Natural Language Understanding component is divided in two part: the NLU-intent and the NLU-slots.

1) NLU-intent

The NLU-intent is responsible for processing user input and performing intent classification. Its main function is to identify and separate different intents within the user's input.

The output of this component is a JSON object containing the detected intents along with the corresponding text segments that represent each intent.

This component keeps track of the conversation history to better understand what the user means. After testing, I found that saving the last five messages works best. This helps the system figure out the right intent by using not just the latest message but also the context from previous ones.

2) NLU-slot

The NLU-slot component is responsible for performing slot filling. It takes the JSON output from the NLU-intent stage and extracts the necessary details for each classified intent. Through testing, I found that creating separate prompts for each intent significantly improves accuracy, allowing the system to better identify and extract relevant slots from the text.

This component also keeps track of the conversation history, not just to take the context, but also to use past system responses for better understanding. For example, if the system suggests three exercises and later the user says, "Add the first exercise to my favorite list," the NLU-slot can look back at the previous response, find the name of the first exercise, and correctly fill in the new slot.

B. DM

The Dialogue Manager is composed of three main components: the state tracker, the action selector, and the database manager. In this case I used only one prompt for all Dialogue Manager.

1) State Tracker

The state tracker is responsible for processing the information extracted by the NLU and keep updating a list of previously detected intents and slots.

For each extracted intent of NLU, the state tracker checks whether it already exists in the list of all intents and slots extracted. If the intent is already present, it updates the stored intent with any new slot values or modifies the existing ones. If the intent is new, it adds it to the list along with its corresponding slots.

2) Action Selector

The action selector is responsible for the core function of the *Dialogue Manager*, which is selecting the next best action in the conversation.

The system can choose from four possible next actions:

- **request_slot(slot)**: If a required slot is missing (i.e., has a null value), the system prompts the user to provide it.
- **confirmation(intent)**: Once all necessary slots are filled, the system confirms the intent execution with the user.
- **check_info(intent)**: If the Database Manager does not return valid results, the system asks the user to verify or correct the provided information.
- **close_system()**: When the conversation is complete, the system terminates the interaction.

3) Database Manager

The Database Manager is responsible for handling all interactions with the database that is MegaGymDataset [3] from Kaggle [4].

Whenever a confirmation action is triggered, the Database Manager performs an operation based on the detected intent:

- *get_exercise, get_information, or list_favorite*: Filters the database using the slots information provided by the user.
- *get_plan*: Generates a workout plan by selecting three random strength exercises, one cardio exercise, and one stretching exercise for each session requested by the user.
- *save_exercise*: Adds a new exercise to the database.
- *add_favorite and remove_favorite*: chenge favorite field in the database for a specific exercise.

The system returns the extracted data in a structured JSON format.

C. NLG

The final component is the Natural Language Generator (NLG), which is responsible for generating user responses. It formulates replies based on the instructions provided by the Dialogue Manager and integrates relevant information retrieved by the system to ensure accurate and meaningful interactions.

Similar to the NLU component, different prompts are used for different intents to guide LLama in producing the most appropriate responses.

In multi-intent scenarios, where the user requests multiple intents in a single interaction, the NLG first generates a separate response for each intent using LLama. It then performs an additional query to LLama to merge these responses, ensuring a more cohesive and natural reply.

IV. Evaluation

During the evaluation process, each component was tested independently by quantitative and qualitative measurement.

A. Intrinsic Evaluation

Intrinsic evaluation was performed exclusively on the NLU and DM components. For NLU, a test set of 50 user inputs was generated using ChatGPT-4o to ensure diversity. These inputs comprehensively cover all application scenarios, including single-intent and multi-intent cases, out-of-domain inputs, and all supported intents. A similar approach was used for DM, but instead of raw user inputs, the evaluation was conducted using a structured dictionary of intents and slots as input.

Metric	Score
Intent Accuracy	91.38%
Slot Accuracy	95.77%
DM Accuracy	94%

Table 1: Quantitative validation

The intent extractor has the lowest performance among the evaluated components, as it handles the most complex task. Identifying intents can be particularly challenging when they are not explicitly mentioned in the user input. Despite this, it achieves an accuracy of 91%, which is a strong result (see Table IV to see score for each intent).

The slot extractor in NLU performs well, demonstrating high accuracy in extracting relevant details from user inputs. Similarly, the Dialogue Manager (DM) was validated by ensuring that extracted slots did not contain null values and that the selected next-best action aligned with expected logic. For instance, the system correctly triggers a request for missing information when required slots are empty.

B. Extrinsic evaluation

To evaluate the system's performance more effectively, I designed a Google Form with five questions, each rated on a scale from 0 to 5. However, due to the limitation that the program could only be tested on my personal computer, I was able to gather feedback from only three users. The collected responses can be found in the Appendix (see Table V).

Overall, the system performs well and provides a good user experience. However, some issues were observed, particularly with the intent extractor. LLama does not always correctly identify the intended user request, and when an intent is misclassified, the error propagates through the pipeline, ultimately affecting the Natural Language Generation (NLG) component. As a result, the system may generate incoherent responses.

Another notable issue is response speed. Since LLama is computational hungry, users found the system somewhat slow, which could impact overall usability.

V. Conclusion

LLamaFit demonstrates strong potential as a human-machine dialogue system for personalized workout planning and exercise recommendations. The system effectively handles a variety of user requests, including full workout plans, specific exercise suggestions, and managing favorite exercises. Through intrinsic and extrinsic evaluations, it has been shown that the NLU and DM components perform well, with high intent accuracy (91.38%) and slot accuracy (95.77%), ensuring effective intent recognition and structured responses. However, some challenges remain, particularly in intent classification errors, which can propagate through the pipeline and lead to misaligned responses in the NLG component. Additionally, Ollama's response time is slow, impacting user experience. Despite these limitations, LLamaFit provides a functional and user-friendly experience, with a well-structured conversational model that successfully guides users through their fitness queries. Future improvements should focus on enhancing intent classification accuracy, optimizing response speed, and expanding user testing to gather more diverse feedback. Overall, this project highlights the feasibility of leveraging LLMs for fitness assistance.

APPENDIX

TABLE I: Intent and Related Slots

Intent	Slots
get_exercise	`type` (str) - Type of exercise (e.g., strength, cardio, stretching) `body_part` (str) - Targeted muscle group (e.g., chest, glutes) `equipment` (str) - Required equipment (e.g., bodyweight, bands, weights) `level` (str) - Difficulty level (e.g., beginner, intermediate, advanced)
get_plan	`level` (str) - Difficulty level (e.g., beginner, intermediate, advanced) `n_session` (int) - Number of sessions per week
get_information	`title` (str) - Name of the exercise `body_part` (str) - Targeted muscle group (e.g., chest, glutes)
save_exercise	`title` (str) - Name of the exercise `description` (str) - Execution description `type` (str) - Type of exercise (e.g., strength, cardio, stretching) `body_part` (str) - Targeted muscle group `equipment` (str) - Required equipment `level` (str) - Difficulty level `duration` (int) - Duration of the exercise (minutes)
add_favorite	`title` (str) - Name of the exercise
remove_favorite	`title` (str) - Name of the exercise
list_favorite	`type` (str) - Type of exercise (e.g., strength, stretching, cardio) `body_part` (str) - Targeted muscle group (e.g., chest, glutes)
give_evaluation	`evaluation` (number) - Numeric rating for session feedback `feedback` (str) - User comments or improvement suggestions

TABLE II: This table presents the intents in the LLamaFit system along with their respective slots. Most intents include attributes that define user preferences, workout details, or interaction requirements. The ‘get_exercise‘, ‘get_plan‘, and ‘save_exercise‘ intents contain multiple parameters to ensure personalized recommendations, while ‘list_favorite‘ retrieves saved exercises based on filters. The ‘remove_favorite‘ intent allows users to delete saved exercises from favorite list. The ‘add_favorite‘ intent allows users to add an exercise to favorite list.

TABLE III: NLU Intent-wise Performance

Intent	Precision	Recall	F1-score
get_information	1.00	0.82	0.90
get_exercise	1.00	1.00	1.00
get_plan	1.00	1.00	1.00
save_exercise	1.00	1.00	1.00
add_favorite	1.00	1.00	1.00
remove_favorite	1.00	1.00	1.00
list_favorite	1.00	1.00	1.00
give_evaluation	1.00	0.75	0.86
out_of_context	1.00	0.50	0.67

TABLE IV: This table presents the performance of the NLU intent classification model, evaluated in terms of precision, recall, and F1-score for each intent. Most intents achieve perfect scores, demonstrating high accuracy in classification. However, ‘get_information‘, ‘give_evaluation‘, and ‘out_of_context‘ show lower recall, indicating that some instances were missed.

Timestamp	Overall Satisfaction	Input Understanding	Response Consistency	Clarity of Expression	Response Speed
user 1	4	4	5	5	2
user 2	5	4	5	4	2
user 3	4	4	5	5	2

TABLE V: LLamaFit User Survey Results. The questions asked in the survey were: (1) Overall Satisfaction: How satisfied are you with the LLamaFit system overall? (2) Input Understanding: How well does the system understand your input? (3) Response Consistency: How coherent are the system’s responses with your requests? (4) Clarity of Expression: How well does the system express its responses? (5) Response Speed: How fast is the system in providing answers?

REFERENCES

[1] H. Touvron, A. Jiang, I. Radić *et al.*, "Llama 3: Open foundation and fine-tuned chat models," Meta AI Blog, 2024, accessed: 2024-02-11. [Online]. Available: https://ai.meta.com/llama

[2] O. Team, "Ollama: Run large language models locally," Official Website, 2024, accessed: 2024-02-11. [Online]. Available: https://ollama.com

[3] VirtualCrush, "Megagymdataset preprocessing and recommendations," Kaggle Notebook, 2024, accessed: 2024-02-10. [Online]. Available: https://www.kaggle.com/code/virtualcrush/megagymdataset-preprocessing-recommendations

[4] Kaggle, "Kaggle: Your machine learning and data science community," Website, 2024, accessed: 2024-02-10. [Online]. Available: https://www.kaggle.com/