Human-Machine Dialogue Project Report

Luca Cavaliere (264386), Francesco Sorrentino (256151)

University of Trento

luca.cavaliere1@studenti.unitn.it, francesco.sorrentino@studenti.unitn.it

1. Introduction

The aim of this project is to create a video-game shopping assistant that interacts by chatting with the user. Instead of navigating menus or filling out rigid forms, users can express what they want in their own words. The assistant interprets their requests, retrieves the relevant information, and responds in a clear and context-aware way. To achieve this, the system is organized as a modular dialogue pipeline. Each component has a well-defined role, which makes the whole system easier to understand, evaluate, and improve.

1.1. Dialogue System Description

1.1.1. Concept and motivation

The main idea behind this project is to offer a more intuitive and human-friendly way to explore video-game information. Rather than forcing users to type precise keywords or manually filter long lists of results, the assistant lets them speak naturally:

- *"Find games similar to Hades."*
- *"Which one is cheaper, Elden Ring or Cyberpunk 2077?"*
- *"Give me the review of Dead Cells."*

The conversational interface has some advantages:

- **Accessibility** - users can interact with the system without any need to learn how to use a new application interface, just with natural language.
- **Engagement** - the interaction is much more engaging for a user.
- **Control** - the modular system gives more control to the developer and allows interdependent benchmarking and improvement of each component.

The responses also are designed to be concise and polite in order to satisfy the user. A critical ethical aspect is also that the system should not decide for the user but simply help the user by suggesting a possible purchase. This is crucial for example when comparing different products.

1.1.2. Types of exchanges, inputs and outputs

The system can react to a wide variety of requests in the videogame domain. The user is able to request for information regarding videogames and gaming terminology. The assistant also provides the user with real-time reviews if requested to motivate or discourage a purchase. The interaction are also tailored to the user because the system remembers the user's favourite games and their friends games that might be helpful to start a new conversation. To keep the dialogue task-oriented the system can also handle out-of-domain requests gracefully with a fallback policy. Furthermore error-recovery is also present in order to continue the conversation seamlessly. In order to handle all of these requests the system can interact with the Steam Games 2025 Dataset[1], the online steam reviews API[2] and a glossary of gaming terms extracted from Wikipedia[3].

1.2. User Group

The system is designed for a variety of users, from more experienced gamers to newcomers that just want to browse the store. The accessibility of the system was the main focus, allowing use also to unexperienced beginners that might not know what some terminology might mean or how to find a game that fits specific archetype. The system is also design with being able to handle different ways to phrase the requests to adapt to any possible customer.

2. Conversation Design

2.1. Dialogue Types

The project requires the study and handling of different varied interactions:

1. **Information retrieval**: The user asks the system for some information regarding either a game or a gaming term (for example "RPG" or "level"). To extract the required knowledge the assistant will query a knowledge base to gather the information and lexicalize it for the user. If the request includes reviews, they are retrieved from the latest available and the system performs sentiment analysis before presenting the results to the user.
2. **Finding games**: The user request for some new games to play that have some characteristics. The system will then use these characteristics to query the knowledge base and present the user with from 2 to 5 options.
3. **Comparing games**: User asks the system to make a comparison of two games based on a criteria ("price", "genre" and "review"). The system will then query the knowledge base and present to the user the similarities and differences of the two games.
4. **Storing user preferences**: The agent also has access to a wishlist where the user can save his favorite games. The user may also access to their friends wishlist to get an idea for a gift or for the next game they will play.

The choice of delving deeper into these specific dialogue types was motivated by observing and studying different forum threads where users were finding the next game to play.

2.2. Interaction properties

To handle different types of dialogues, It is not enough to simple present the answer the user wants. The agent must also be engaging and use a set of strategies to enhance the interaction.

2.2.1. Initiative

The agent uses mixed initiative to try making the interaction more natural and also to reach the goal faster. This is done by

either proactively proposing options to the user e.g. "Do you want to compare games by price, genre or reviews?" and by suggesting what they could do next e.g. "Would you like to search for information on one of these games or adding one to your wishlist?".

2.2.2. Acknowledgment

The system uses explicit acknowledgment to inform the user that their request has been understood. This is done by tracking the dialogue state constantly and injecting it into the response.

2.2.3. Confirmation

The system asks if the user is happy with the proposed games, leaving to the user the possibility of asking for different ones.

2.2.4. Fall-back responses

If the user inserts gibberish or out of domain request they are properly handled by warning the user. Other fall-back responses are used when the user asks for non existing objects in the database, e.g. asking for game "abc" when game "abc" is not found in the database.

2.2.5. User profile

The system is able to handle different possible users: coherent, incoherent, over-informative and under-informative. When necessary the systems asks the user for clarification if not enough information is provided to fully complete a task.

2.2.6. Multiple intents

The system also has the advantage of being able to handle two intents at the same time. However if the number of intents detected is more than 2, the system will then warn the user and attend only to the first request. This has been done to adapt to even more types of user's requests while not responding with over-informative and unclear answers.

2.2.7. Conversational markers

The agent uses markers like "Perfect", "Fantastic" and "Great". This ensures the conversation does not feel robotic and is more engaging for the user.

2.2.8. Error recovery

The system also handles cases when external errors occur by warning the user. For example, if the extraction of real-time reviews fails the system will warn the user and tell him to try again later.

2.2.9. Conversation history

The system also has a window of memory of the conversation and is able to remember implicit references to the previous turns. This feature allows for a more natural interaction.

3. Conversation Model

The system is composed of five modular components each with the goal of handling a different stage of the interaction. The idea behind a modular paradigm is that each component is able to be evaluated independently, identifying errors or pitfalls more clearly. This also enables to evaluate different backbones (Table 2) for each component to find the best one for that task. The components used are:

- **Preprocessor**: A preprocessor component that given an input utterance has the goal of splitting it based on the system intents. This allows every split input sentence to only map to one intent.
- **NLU**: The Natural Language Understanding component has the task of extracting intent and slots that form the dialogue state from a given utterance.
- **DM**: The Dialogue Manager, using the Dialogue State outputs the next action of the agent from the dialogue state.
- **NLG**: The Natural Language Generator, has the goal of lexicalizing the outputs of the other components and the knowledge extracted.
- **SA**: The Sentiment Analysis component is used to parse store reviews and collect their sentiment to report it to the user.

These components are combined to form the pipeline that processes the user prompt.

3.1. Pipeline

Figure 1 diagram shows how the different components are connected and used in the interaction. The input first is parsed by the Preprocessor that splits it into sentences for each intent. Based on the number of intents there are three possible scenarios:

1. If the user intent is only one the processing goes on.
2. If the intents are two the system handles both by joining the two responses.
3. If the intents are more than two, the system adds a flag *multiple_intents* to the NLG input to warn the user that there are too many requests, e.g. "Let's do things step by step". The system will then attend only to the first request made by the user.

After the first preprocessing step, the NLU will extract the intent and slots from the input utterance. This is then fed into the Dialogue State Tracker (DST) component that stores and updates the Dialogue State of the system. The current dialogue state is then used as an input to the DM that outputs the next best action for the assistant. Based on the action, the necessary knowledge is extracted to be injected into the final response. If the user requested for reviews they are requested through an API call and analyzed by the SA component. The NLG input is composed of the dialogue state, next best action, extracted knowledge and the multiple_intents flag to used to generate the final response. In this overview of the pipeline it is clear why all components must function properly to avoid errors and misunderstandings in the interaction. The main bottleneck performance-wise is the the NLU step that takes the most time, given the complexity of the task. A strategy that is used in the pipeline is also to give the NLU component history of the conversation in order to extract slots from previous exchanges, this however has the cost of increasing the time to complete the extraction task.

3.2. Prompts

The prompts for the LLM backbone of each component all followed a similar structure:

- ROLE: role of the component.

- INPUT: expected input of the component.
- OUTPUT: expected output format of the component.
- RULES: general rules the LLM has to follow.
- CONTEXT: contextual information, for example the list of intents or possible actions
- EXAMPLES: few-shots examples for the component to improve performance.

The prompt-engineering technique of including few-shot examples was used in order to improve performance and robustness of the models. For the DM and NLG component the prompt were also customized for each current intent, being able to adapt depending on the current request. This idea's benefits are twofold. First making the generation faster since less context needs to be parsed by the LLM. Secondly the prompt can be tailor-made for a specific situation reducing confusion.

3.3. Intents and slots definition

The task of the NLU component is undeniably very complex and fundamental for the understanding of the user's request. Table 1 shows the definition of the slots and intents that the system has to classify and extract. A wrong definition can cause incorrect behavior of the system and drop the NLU performance. In order to properly define intents and slots, various online conversation on the topic of choosing a game were studied as reference. An important detail is that, when defining intents, It is better to create a single intent instead of many small attribute specific ones, to avoid hallucination and confusion from the LLM. In this project, the use of a single *get_game_info* intent instead of many *get_game_*{*attribute*} proved to improve the robustness and performance of the model. Another observation can be made on the *discover_game* intent where a high number of slots can be observed. The high slot number greatly increases the difficulty when extracting a slot from an utterance because It can cause ambiguity in the LLM, for example between the concept of *developer* and *publisher*. A possible solution to the issues discussed would be to create two NLU components: one that classifies intents and one that extracts slots based on the extracted intent, however this idea introduces a new component to evaluate and an overall heavier pipeline.

4. Evaluation

After the design phase of the agent, the backbone for each component must be evaluated to choose the best alternatives. Table 2 shows the four LLM checkpoints from hugging face hub that where selected for evaluation. A Rule-based alternative was also implemented for the DM component to be compared with the LLM option. This was done given the fact that the DM component handles structured data only and no natural language. The first phase to select the backbone and reach better performance is the so called intrinsic evaluation to be then followed by the extrinsic one.

4.1. Data description and analysis

Data needed to be generated in order to properly evaluate the various component. The data was generated with the aid of the Gemini Pro model.

4.1.1. NLU data

From two to three templates were generated for each intent with slots to be filled in. The maximum number of slots to be filled in per template was set to four slots to simulate a realistic scenario of a user searching for a game. After generation the templates were validated to ensure consistency with intent definition. This strategy enabled to obtain a large amount of data that did not require a long validation phase. The slots were filled in with all the possible values in case of slots with a close group of values (genre, mode, ...) and with some random values extracted from the knowledge base in case the slots had an open set of values (title, name,...). The resulting data amounted to 980 samples with roughly 40% samples destined to the more complex intent *discover_game*.

4.1.2. DM data

The data for evaluating the DM component was generated with no LLM and just templates for each possible dialogue state. Then a rule-based function was implemented mapping from dialogue state to action. The slot values used to fill in the templates was the same as in the NLU data generation step. The final result was 1588 samples with roughly 40% samples for the *discover_game* intent that has the highest amount of slots.

4.1.3. NLG and Preprocessor data

The generation of the data for the NLG and Preprocessor followed the same idea. The LLM was used to generate the utterances based on the structured NLG input for that component. For the Preprocessor first user input were generated and then they were manually split based on the systems intents. The data was validated to ensure the LLM output can be used for the evaluation without introducing bias. The final data was limited to around 100 samples for each component given the amount of human work involved in the generation.

4.1.4. SA data

To generate data for the SA review-like text was generated from the LLM and manually annotated as either 'positive', 'negative' or 'neutral'. The generated text was validated to ensure that it followed the input prompt and was similar enough to expected review text. The final sample count was 500 samples distributed evenly among the three sentiment labels.

4.2. Intrinsic evaluation description

The instrinsic evaluation consited of collecting the predictions of the variuous components and computing the automatic metrics. The chosen metrics for each component were:

- NLU $\rightarrow$ Intent Accuracy and Slot F1 score.
- DM $\rightarrow$ Accuracy.
- NLG $\rightarrow$ F1 score and BLEU score.
- Preprocessor $\rightarrow$ F1 score and BLEU score for each split sentence.
- SA $\rightarrow$ Accuracy.

The metrics were computed for each backbone in table 2 for each component. The automatic metrics for the NLG and Preprocessor were not found to be informative on the actual quality of the component, so the final evaluation was done by directly inspecting and comparing the outputs of the various different backbones. The comparison has been done with the same prompt and data in order for it to be fair. The only difference in approach is that gemma needs user prompt to be merged with the system prompt, and does not output json directly but by marking it with 'json' tags.

4.3. Extrinsic evaluation description

The human evaluation has the goal of evaluating the real capabilities of the agent. The task the candidates had to fulfill was interacting with the system in order to find a game for a friend. This task was defined in order to test most of the functionalities of the system while not being too open and offering a clear goal to the user. After the interaction the candidates had to evaluate the system on 4 criteria with a score from 1 (Completely disagree) to 5 (Completely agree):

Understanding When asking the system to search games or comparing them, did it follow your constraints correctly and complete the task appropriately?

Management Did the system ask properly for further information or clarification when the task was unclear?

Generation Were the system's responses grammatically correct, polite and concise?

Usefulness Would you consider using the system again in the future?

If necessary, the candidates were also able to explain their decision with less than 200 words. The description of some of the criteria took inspiration from the previous work of Mousavi et Al.[4]. The candidate requirements were to have a good knowledge of the english language (at least a B2 certification) and also some minimal knowledge on the videogame domain.

4.4. Results of intrinsic evaluation

The tables 3,4 and 5 show the results of the intrinsic evaluation for the NLU, DM and SA. The performance of the intent classification was higher when the qwen3 model was chosen, with the mistral model being the second best for slot filling. The final choice was the qwen3 model that was slightly faster than mistral. The DM evaluation showed that the best LLM performance is still with the qwen3 model but the absolute best performance is with the rule-based strategy that is also much faster. For this reason, the rule-based strategy was chosen for the final system configuration. The best sentiment analysis component is qwen3 with perfect score on the test set with llama3 as a close second and so was chosen for the final pipeline. The NLG evaluation took into account all interaction properties in order to choose the best model. The gemma and mistral backbones were found to not be proactive enough, being very concise. Heuristically, llama3 seemed to yield the best responses, respecting the dialogue properties the agent was designed with. Finally, for the preprocessing component, the qwen3 model showed to be the most reliable in splitting text without altering it, and therefore was chosen for the final model configuration.

4.5. Results of extrinsic evaluation

The final results of the evaluation can be seen in table 6 and was conducted by 11 participants. Analyzing the values, the final agent was able to guide the candidates with the task while keeping the interaction engaging and polite. The agreement between candidate's scores is high (std < 1) and shows that there was likely no phenomenon of randomly picking a score. The scores are generally very positive and show how the system capabilities satisfied the users. An important remark is that the candidates tested the system using a minimalist GUI in order to better read the system outputs and simulate a final deployment. However, this could have indirectly influenced their evaluation by focusing on the visual aspect and not the actual dialogue of the agent. Some candidates noted how an even more flexible system could be an improvement when searching games, for example choosing a game from a gameplay element or from the name of the main character.

4.6. Errors and limitations

The system has several limitations and errors where improvements can be made. A rare NLU error happens when the model outputs incorrect json objects, e.g. missing commas or brackets, this is clearly caused by the unpredictability of the LLMs. Another issue that shows how the use of an LLM can cause a lack of robustness is an error in the NLG output: randomly the output will not be proactive or ask further questions. To solve this limitation prompt engineering has been done on the prompt but the agent may still lack proactivity in certain interaction. This issue is linked to the choice of the backbone model and possible solutions would be evaluate other ones that are more proactive or even to add another component to ensure different interaction properties by rephrasing the NLG output.

5. Conclusion

The project shows how LLMs can be leveraged to create a functioning dialogue agent. The agent is able to interact with the user in an engaging and polite manner while helping them to fulfill their final goal. However there are several limitations that come from the use of LLMs for a dialogue agent. One of the main problems is the unpredictability of the LLM backbones , that, even with prompt-engineering, sometimes do not return the expected output or follow the rules declared in the system prompt. Another issue linked to prompt engineering is that the developer does not know what effect changes to the prompt might have on the model and so there is no clear way to improve performance other than trying to evaluate many variations of the prompt to find the best one. Other solutions to further improve the system would be to add other components after the NLG to improve the output, for example by rephrasing it or making it more proactive for the user. Another interesting idea would be to split the NLU module in two with the first block classifying intents and the second that extracts the slots with the information of the current intent. The issue with both of these ideas is that the new components need to be evaluated and they also decrease the time to generate a response, so they are left to future work.

6. References

[1] A. Ermilov, A. Nevolina, A. Pospelov, and A. Kubaeva, "Steam games dataset 2025," 2025. [Online]. Available: https://www.kaggle.com/dsv/11017460

[2] Valve Corporation, *User Reviews — Steamworks Documentation*. [Online]. Available: https://partner.steamgames.com/doc/store/getreviews

[3] Wikipedia contributors. Glossary of video game terms. Wikipedia, The Free Encyclopedia. [Online]. Available: https://en.wikipedia.org/wiki/Glossary_of_video_game_terms

[4] S. M. Mousavi, G. Roccabruna, M. Lorandi, S. Caldarella, and G. Riccardi, "Evaluation of response generation models: Shouldn't it be shareable and replicable?" in *Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*. Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, Dec. 2022, pp. 136–147. [Online]. Available: https://aclanthology.org/2022.gem-1.12

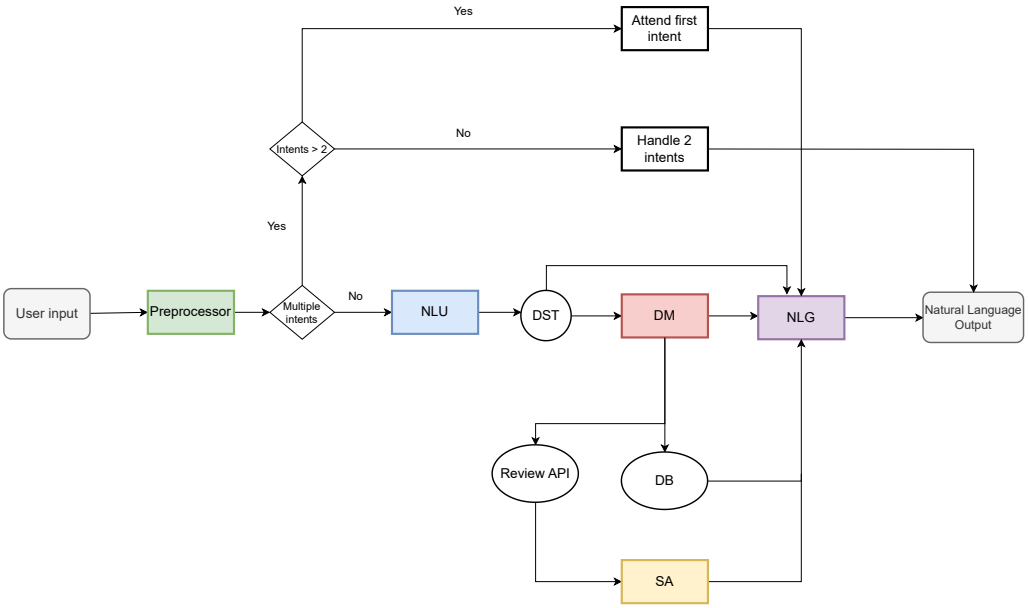

Figure 1: *The architecture of the dialogue agent. The pipeline of the components allows the system to generate the response for the user. Multiple intents are also handled. An external SA component is used to analyze reviews for insightful information.*

Intent	**Slots**
get_game_info	*title* (str): Title of the game to search info on. *info* (str): Info requested [summary, genre, mode, platform, required_age, review, price].
discover_game	*genre* (str, opt): Values [action, adventure, rpg, strategy, etc.]. *price* (float, opt): Higher bound price to use for search. *release_year* (int, opt): Year of release (e.g., 2015). *platform* (str, opt): OS where playable [windows, mac, linux]. *mode* (str, opt): Values [singleplayer, multiplayer]. *similar_title* (str, opt): Another game to find a similar one. *required_age* (int, opt): Age required to play. *publisher* (str, opt): Publisher of the game. *developer* (str, opt): Developer of the game.
compare_games	*title1* (str): First game to compare. *title2* (str): Second game to compare. *criteria* (str, opt): Values [price, reviews, genre].
get_friend_games	*name* (str): Name of the friend.
get_term_explained	*term* (str): Term to be explained (e.g., RPG, MMO).
add_to_wishlist	*title* (str): Game to add to wishlist.
remove_from_wishlist	*title* (str): Game to remove from wishlist.
get_wishlist	–
out_of_domain	–

Table 1: *Intents and slots defined from the studied interactions.*

Model Name	**Hugging Face Checkpoint**
qwen3	Qwen/Qwen3-4B-Instruct-2507
mistral	mistralai/Mistral-7B-Instruct-v0.3
llama3	meta-llama/Meta-Llama-3.1-8B-Instruct
gemma	google/gemma-2-9b-it

Table 2: *Checkpoint evaluated in the project for every component.*

Model	Intent Acc.	Slot F1
qwen3	0.99	0.95
mistral	0.99	0.96
llama3	0.95	0.93
gemma	0.97	0.96

Table 3: *NLU results for the various backbones. In **bold** the model that was chosen for the final agent.*

Model	Accuracy
qwen3	0.93
mistral	0.68
llama3	0.82
gemma	0.83
Rule-Based	1.00

Table 4: *DM results for the various backbones. In **bold** the one chosen for the final model.*

Model	Accuracy
qwen3	1
mistral	0.88
llama3	0.99
gemma	0.95

Table 5: *SA results for the various backbones. In **bold** the one chosen for the final model.*

Criterion	Score
Understanding	4.3 ± 0.4
Management	3.9 ± 0.8
Generation	4.5 ± 0.5
Usefulness	4.3 ± 0.6

Table 6: *Human evaluation results of 11 candidates. Mean and std scores of the candidates show the agreement between evaluations.*

Intent Type	Accuracy
get_game_info	1.0000
discover_game	1.0000
compare_games	1.0000
get_friend_games	0.8125
get_term_explained	1.0000
add_to_wishlist	1.0000
remove_from_wishlist	1.0000
get_wishlist	1.0000
out_of_domain	1.0000
Overall Accuracy	**0.9969**

Slot Type	F1-Score	Precision	Recall
info	0.9091	0.9091	0.9091
title	0.9748	0.9748	0.9748
required_age	0.9181	1.0000	0.8486
developer	0.9138	0.9953	0.8446
price	0.9575	0.9978	0.9203
mode	0.9111	1.0000	0.8367
platform	0.9786	1.0000	0.9582
release_year	0.8832	1.0000	0.7908
similar_title	0.9398	1.0000	0.8865
publisher	0.9288	0.9954	0.8705
genre	0.9746	0.9938	0.9562
title1	1.0000	1.0000	1.0000
title2	1.0000	1.0000	1.0000
criteria	1.0000	1.0000	1.0000
name	0.8966	1.0000	0.8125
term	1.0000	1.0000	1.0000
Overall Weighted	**0.9462**	**0.9965**	**0.9008**

Table 7: *Fine-grained details of the qwen3 NLU performance. The NLU component confused some templates to view the friend wishlist with out_of_domain ones lowering accuracy. Also the slots of the discover_game intent are the ones with lowest accuracy as expected-*