Use of a generative chatbot as a middleman to improve User Experience in Interactive Fiction games

Adriano Tolfo Dotta[1], **Marcelo Resende Thielo**[1], **Jean Felipe Patikowski Cheiran**[1]

[1]Campus Alegrete - Universidade Federal do Pampa (UNIPAMPA)
Av. Tiarajú, 810 – 97546-550 – Alegrete – RS – Brazil

{adrianodotta.aluno,marcelothielo,jeancheiran}@unipampa.edu.br

Abstract. *In the 1970s and 1980s, interactive fiction games emerged and changed the gaming world by enabling players to talk to the machine, make their own decisions, choose their paths, and decide what to collect and do. This freedom captivated players, but at the time, games were limited by programmed scripts that only accepted words present in the game's code dictionary in a limited format, usually two or three keywords by phrase. However, technological advances have paved the way for improvements in this regard, replacing scripts with artificial intelligence using APIs, such as ChatGPT, for example. Thus, it may be possible to offer players an even more natural and customized experience, where each decision is more flexible with the player's writing style. In this work, we carried out a communication experiment with the ChatGPT as an intermediary for interactive fiction games and the Frotz game interpreter (a modified version of the Z-Machine interpreter) to assess the feasibility of the approach.*

1. Introduction

In recent years, the emergence of ChatGPT as an intelligent tool has opened up significant possibilities for improvement in various areas of knowledge [Wu et al. 2023]. One of these areas is digital gaming, where this technology has proven to be a valuable ally in pursuing an increasingly immersive and personalized experience for players [Biswas 2023]. A specific game genre that closely relates to natural language processing (NLP) and its technologies, whose user interface is frequently based exclusively on text, is the one known as interactive fiction [Montfort 2011]. In this context, an aspect that has garnered attention is that interactive fiction games often offer limited player interaction with the virtual environment, requiring phrases limited on containing just a pair of words in the format `<verb subject>`. Even though, they are still venerated by players that consider them imagination-stimulant and challenging. To overcome this limitation, ChatGPT emerges as an excellent tool for use as an intermediary input parser, allowing players to type commands with much more flexibility without being restricted by predetermined rules of very limited parsers and assisting them in more complex situations. In this regard, exploring the possibilities of using ChatGPT as an interpreter of user inputs in interactive fiction games is relevant, aiming to provide an even richer and more interactive experience for players and maybe give this game genre a revamp.

The main objective of this study is to examine the feasibility of using the ChatGPT API as an interface to enhance the user experience of interactive fiction games, thereby simplifying gameplay and providing players with greater freedom.

2. Related work

Even though ChatGPT is considered a promising test bed for studying language-based autonomous agents [Hausknecht et al. 2020], few references describe the use of ChatGPT in the context of interactive fiction games. In addition to guides and tutorials to create text-based adventure games by connecting ChatGPT API to programming languages [Trivedi 2023], some work-in-progress reports detail the use of ChatGPT and other large language models to play text-based games.

Tan and colleagues carried out experiments using FLAN-T5, Turing, and OPT language models to solve puzzles from the Detective game [Tan et al. 2023]. The results show a significantly lower performance of the three language models compared to the human and deep reinforcement learning model performances.

Also, Tsai et al. used ChatGPT to play the Zork I game using a human as an interface [Tsai et al. 2023]. While the authors state that the performance of ChatGPT is promising, the model was unable to keep a consistent world model of the game (ChatGPT has even hallucinated when asked to recall previous locations) or infer the goals of the game by itself.

3. Proposal

Using the Frotz interpreter demands a specific procedure: after installation, a command line specifying the desired game's name must be executed in the terminal that starts the game and displays the initial text describing the environment. Additionally, a text input field is provided for player interactions. The proposed solution involves integrating the OpenAI Chat API into this text input field, allowing the entered text to be analyzed and converted into game script commands.

3.1. Technological overview

Initially, we assessed the availability of the API [OpenAI 2020], along with associated costs. Among the options, our primary focus for this study is the exclusive "Chat" feature (gpt-3.5-turbo) to be used in a client-server implementation between C and Python programming languages. Regarding availability and cost, this choice proves to be feasible with minimal expenses, amounting to $0.0015 per 1000 input tokens and $0.002 per 1000 output tokens. Additionally, for interpreting these games and integrating the ChatGPT API into the interpreter's code, we used the Frotz interpreter [Griffith and Jokisch 2016], which is compatible with several platforms, e.g., Windows, Linux, DOS, and others. While a first implementation attempt was made with the Windows version of the software, we could not get it working because, although we have followed the step-by-step guide provided by the author strictly, many compiling errors for the open-source version were still obtained. As a result, we opted to build the Linux version of the engine named "Dumb Frotz", which is a stripped-down variant of Frotz interpreter that preserves the original user experience without WIMP interfaces, colors, and sounds. This command-line interface was the most suitable option for carrying out our tests.

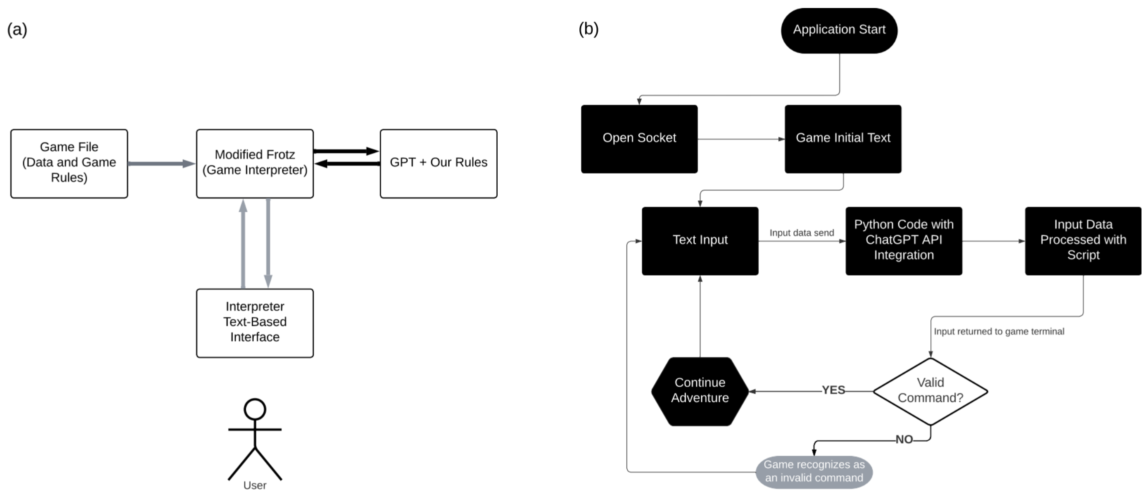
3.2. Model overview

To facilitate this integration, we opted for a client-server architecture, where the Frotz code (written in the C programming language) communicates with a Python codebase containing the ChatGPT integration. Python was chosen for its simplicity of implementation and better compatibility with API calls compared to the efforts of integrating and invoking APIs in the C programming language. Upon launching the selected game, a communication socket is established to transmit the text input field information to the Python code, where the received text is analyzed concerning the relevant game code. It is important to acknowledge that this procedure may encounter challenges due to the ChatGPT's response pattern, which often includes additional contextual information or explanations. To mitigate this issue, we implemented a message format rule in the code to ensure that the returned response consistently follows a command in the `<verb subject>` pattern. An example of an implemented rule is structured as follows[1]:

```
If the following sentence contains the words 'pick, get,
collect or hold' (verb list), and also the word 'helmet'
(target) at any point, return to me directly only the
phrase 'get helmet' (verb + target), without any additional
words, explanations, or anything else, just as requested
within quotation marks.  The sentence is:  [text input]
```

Following the analysis, the sentence provided by the user to the Frotz text interface will be sent to GPT that, already knowing our parsing rules, will return the formatted command (verb and subject) to Frotz that will deliver the game output to the user interface. As a result, even if multiple words are entered in the input field, only the crucial keywords to the command will be considered, minimizing the occurrence of "I don't understand that sentence." from the game engine. This specific command returned to the interpreter will provide a more natural, dynamic and adaptive text input without extensive modifications to the source code. The preliminary system architecture and the execution flowchart are both shown in Figure 1.

Figure 1. (a) Preliminary system architecture (darker arrows indicate data exchanging through socket connection; lighter arrows consist in original unmodified relations of the system), and (b) Execution flowchart.


(a)
Game File (Data and Game Rules)
Modified Frotz (Game Interpreter)
GPT + Our Rules
Interpreter Text-Based Interface
User
(b)
Application Start
Open Socket
Game Initial Text
Text Input
Input data send
Python Code with ChatGPT API Integration
Input Data Processed with Script
Input returned to game terminal
Valid Command?
YES
Continue Adventure
NO
Game recognizes as an invalid command

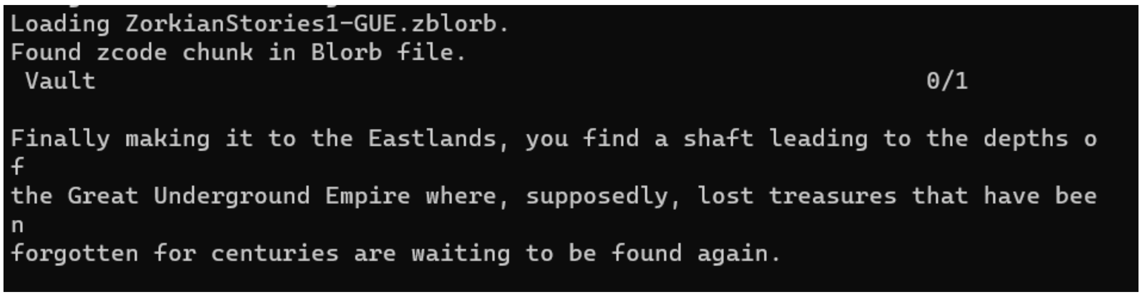
[1]The original pattern was written in Brazilian Portuguese but translated into English for this paper.

4. Early results

This section presents some of the results we obtained in our experiments so far. We set up the environment with the Frotz interpreter compiled with the socket library and the Python code with the ChatGPT API running to enable client-server communication and message parsing.

Figure 2 presents the Zorkian game loading and running in the modified Z-Machine interpreter (Frotz).

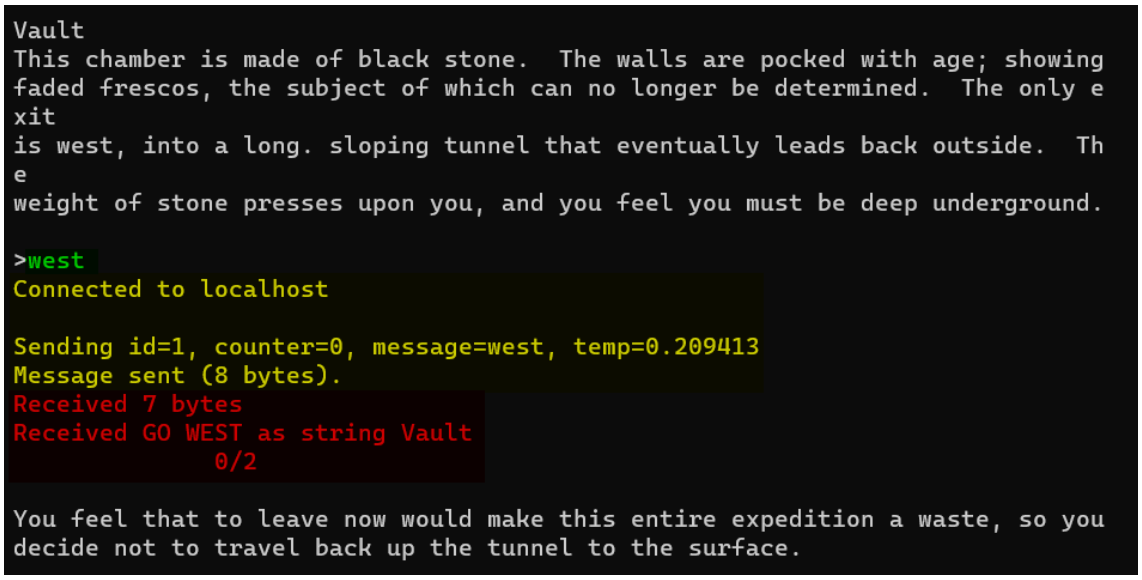
Figure 2. Zorkian game running in the Frotz interpreter.



```
Loading ZorkianStories1-GUE.zblorb.
Found zcode chunk in Blorb file.
 Vault                                                           0/1

Finally making it to the Eastlands, you find a shaft leading to the depths o
f
the Great Underground Empire where, supposedly, lost treasures that have bee
n
forgotten for centuries are waiting to be found again.
```

Figure 3 shows the communication ongoing in our prototype: (1) the user input message (green) is intercepted by our client feature added to Frotz source code and sent to the ChatGPT API server running in Python; (2) the socket connection log information (yellow) is visible for debugging; and (3) the translated string (red) received and processed with game unfolding (white).

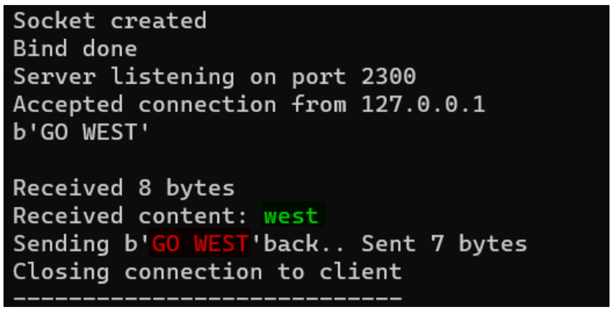
Figure 3. Communication sample between Frotz (client) and ChatGPT API.



```
Vault
This chamber is made of black stone.  The walls are pocked with age; showing
faded frescos, the subject of which can no longer be determined.  The only e
xit
is west, into a long. sloping tunnel that eventually leads back outside.  Th
e
weight of stone presses upon you, and you feel you must be deep underground.

>west
Connected to localhost

Sending id=1, counter=0, message=west, temp=0.209413
Message sent (8 bytes).
Received 7 bytes
Received GO WEST as string Vault
              0/2

You feel that to leave now would make this entire expedition a waste, so you
decide not to travel back up the tunnel to the surface.
```

Finally, Figure 4 demonstrates the server-side Python software showing connection messages, the received string from Frotz (green), the returned message (red), and the connection closed.

Figure 4. Communication sample on the server-side.



```
Socket created
Bind done
Server listening on port 2300
Accepted connection from 127.0.0.1
b'GO WEST'

Received 8 bytes
Received content: west
Sending b'GO WEST'back.. Sent 7 bytes
Closing connection to client
----------------------------
```

5. Conclusions

The results of the early experiments we conducted suggest that it is feasible to integrate an interactive fiction interpreter with a generative chatbot API such as ChatGPT to improve the user experience. Since many players demand more flexible and error-tolerant games, this improvement could increase the game genre's popularity and also foster new reading habits among young players. We expect to incorporate more vocabulary and features in the prototype in the next few weeks and investigate the possibility of using GPT also for transforming the messages of the engine output.

References

Biswas, S. (2023). Role of chatgpt in gaming: According to chatgpt. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4375510.

Griffith, D. and Jokisch, S. (2016). Frotz - a portable z-machine interpreter. https://davidgriffith.gitlab.io/frotz/.

Hausknecht, M., Ammanabrolu, P., Côté, M.-A., and Yuan, X. (2020). Interactive fiction games: A colossal adventure. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7903–7910.

Montfort, N. (2011). *Twisty Little Passages: An Approach to Interactive Fiction*. MIT Press, Cambridge, MA, USA.

OpenAI (2020). OpenAI API. https://platform.openai.com/docs/introduction.

Tan, Q., Kazemi, A., and Mihalcea, R. (2023). Text-based games as a challenging benchmark for large language models. https://openreview.net/forum?id=2g4m5S_knF.

Trivedi, K. (2023). Creating a personalized text-based adventure game with chatgpt api: A step-by-step guide to crafting dynamic narratives.

Tsai, C. F., Zhou, X., Liu, S. S., Li, J., Yu, M., and Mei, H. (2023). Can Large Language Models Play Text Games Well? Current State-of-the-Art and Open Questions. https://arxiv.org/abs/2304.02868.

Wu, T., He, S., Liu, J., Sun, S., Liu, K., Han, Q.-L., and Tang, Y. (2023). A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1122–1136.