Word Explorer:
An AI-Driven Text Adventure Odyssey

Based on NLP

FINAL REPORT

Jiaxu Li, Zhipei Wang, Zhenhao Tu, Yueju Han, Yudong Hu, Dingyu Yang
Marco Palomino
Group 6 Zeta
2024/6/1

I. Executive Summary

This project, aims to create an **AI-driven text adventure game** that provides accessible mental support, a secure emotional outlet, and services to improve the quality of life for individuals under stress. The key objectives includes selecting a suitable NLP model, preparing and preprocessing datasets, designing a user-friendly interface, fine-tuning the model, and deploying it as a functional website.

Objectives:

1. **Model Selection:** Initially consider **META-LLaMA-2** models but shifted to GPT-3.5 Turbo due to platform constraints.
2. **Data Preparation:** Gather and format novel plots, use **GPT-4** for data generation, and applied data augmentation.
3. **UI Design:** Develop a structured and compatible UI with a scrollable chat box and a grid system.
4. **Model Fine-tuning:** Use OpenAI's fine-tuning platform to create a unique deployable model.
5. **Website Deployment:** Implement Flask for backend communication and prompt engineering to enhance user interactions.

Methodology: The project utilized agile methodology for project management, enabling a flexible and responsive approach. Data was collected manually and automatically, followed by rigorous quality checks and data augmentation. The development process involved planning, designing, and implementing both the model and the full-stack website, ensuring scalability and maintainability.

Key Findings:

1. **User Engagement:** The game performed well in engaging players, but improvements are needed in storyline continuity and difficulty balancing.
2. **Challenges:** Resource capacity limitations required switching to OpenAI's model, and generating diverse data efficiently involved using **GPT-4.**
3. **Model Performance:** Addressed overly positive responses by emphasizing negative outcomes through prompt engineering.

Conclusion: The project successfully developed a text adventure game that meets the initial objectives. Future improvements include gathering more data for fine-tuning, optimizing backend logic, and enhancing the UI to further improve player satisfaction and the overall gaming experience. The structured analysis and iterative development approach provide a clear pathway for refining the game elements to better meet user needs.

II. Introduction

The development of today's technologies is definitely a dramatic plot in the history of mankind, but what comes with it is the increasing social pressure, leading to a higher

portion of people suffering from mental illnesses. There are lots of mental health resources, however, individuals might not aware of——or refuse to admit——the underlying fact that they suffered from these illnesses, making it difficult for timely psychological treatment services.

Another problem is that the rapid pace of development and ever upgrading demands for life burdens people with great pressure, making people yearn for a moment to escape from reality to fulfill emotional needs and relax their weary minds. **Consequently**, there is a growing demand for applications that can assist in psychological therapy while serving as a safe, healthy, and stable outlet for emotional and stress release.

This project enables us to provide:

1. **Accessible metal support,** which suggests individuals that refuse to ask for proper therapy can try to address their issues on their own.
2. **Secured place for emotional outlet,** allowing individuals to express their pressure without divulging their private information .
3. **Services which satisfies the emotional needs for individuals,** improve their metal status, hence improve their quality of life.

III. Objectives

1. **Find NLP Model:** At first we decided to pick a suitable model from **Hugging-Face LLM leaderboard,** and the **META-LLaMA-2** models were identified as potential choices. However the low capacity of our development platform blocked us to deploy the model. As a result we moved to **GPT-3.5 Turbo** which performance was far well better than models with only **7B~13B parameters,** while OpenAI provided a usable **server** for developers.
2. **Dataset Preparing and Preprocessing:** The first step was gathering novel plots with different topics from websites as raw data. Next the data was manually formatted to satisfied the model's requirements, and then used **GPT-4** to generate similar data. Finally we removed the duplication and applied data augmentation.
3. **UI Design:** We had designed a well-structured UI with a scrollable chat box implemented to show the text. In order to ensure the compatibility, a grid system was also used to maintain the structure.
4. **Fine-tuning Model:** OpenAI has its own fine-tuning procedure and platform, so the only thing we needed to do was uploading the dataset, splitting train/validation sets to ensure performance, and after a while the platform returned a fine-tuned unique model for us to deploy.
5. **Deployment as a Website:** We used **flask** instead of **node.js** to establish the communication between our website pages and the GPT API. In order to make the user-machine interaction better, we utilized the logic to process user input and did prompt engineering.

IV. Methodology

1. **Project Management Technique**

In our AI-driven software implementation project, we adopted **agile methodology** as our project management technique, which is suitable for the small team management. This approach significantly enhanced our team's responsiveness and flexibility, ensuring the project remained on track and progressed steadily.

For us, we organized the development process into **sprints, short, iterative cycles** focused on delivering specific features, which could help the team members to concentrate on distinct tasks within a defined time frame and manager the time arrangement easily. Each sprint began with a planning meeting where tasks were prioritized and assigned to the team members with clear deadline, **while** the tasks were also re-evaluated to accommodate the dynamic nature of the project's development and any arising changes. **Furthermore,** we also had "daily" stand-up meeting for discussing the problems and challenges encountered in previous day so that we can address the problem in time, avoiding it to become a major issue. **However,** we held the meeting three days a week since we still needed to attend the classes and finished other works. This adjustment maintained the benefits of regular communication while respecting the team's workflow and time constraints.

To keep track of our project, **Trello**, a management system, was employed in our project management process. Its intuitive interface and flexibility enable our teams to implement the agile practices effectively.With **Trello**, we could organize our predefined tasks and processes using cards and boards, where each card represents a task, and boards represent different stages of the project and the deadline. This layout provided a clear visual overview, allowing us to check the requirement of the tasks conveniently and clearly. It also facilitated us to alter our detail action plans and task dynamically according to the development process and requirement changing.

2. AI Model

As mentioned earlier, a core aspect of this project was the use of **large language models (LLMs)** as the primary driving engine behind the entire initiative. **Consequently,** selecting the appropriate LLM had become a critical factor. **Currently,** there are two mainstream methodologies for training a LLM within the software development workflow:

a. **Full Training:** By identifying specific story themes, relevant datasets are independently searched for and compiled. Using existing AI frameworks such as **TensorFlow or PyTorch,** an appropriate training framework is chosen to undergo a complete model training process.

b. **Fine-tuning:** This involves using fine-tuning techniques to train downstream tasks on a suitable pre-trained model.

1) The Selection of Training Approaches:

The selection of training approaches could be influenced by some realistic factors like time constraints, resource availability, and the project's specific needs. We had compared these two approaches, detailing their advantages and disadvantages, and explain why, for our project, fine-tuning a pre-trained model was deemed the most suitable approach.

Full Training:

Advantages:

a. **Customization:** Full training allows for the development of a model that is highly tailored to the specific needs and nuances of the project. This is particularly beneficial when dealing with unique datasets or specialized tasks that pre-trained models may not perform well on.

b. **Control Over Data:** This method provides complete control over the data used for training.

Disadvantages:

a. **Resource Intensive:** Starting from scratch requires substantial computational resources. Training a large language model from the ground up demands powerful CPUs and GPUs, which can be prohibitively expensive.

b. **Time-Consuming:** The process of collecting, cleaning, and preparing a bespoke dataset is challenging and lengthy. Moreover, the training process itself can take weeks or even months, which may be not feasible for projects with tight deadlines.

c. **Expertise Required:** Full training necessitates a deep understanding of model architectures, hyper-parameter tuning, and training processes, requiring skilled data scientists and machine learning engineers.

Fine-tuning:

Advantages:

a. **Efficiency**: Fine-tuning a pre-trained model is significantly faster than training a model from scratch. As the model has already learned a considerable amount of general knowledge, only the final layers typically need retraining to adapt to the specific task at hand.

b. **Cost-Effective:** It uses fewer resources, as the most computationally expensive parts of training are already done.

c. **Accessibility:** Fine-tuning can be performed with less specialized knowledge compared to full training. Frameworks and tools that facilitate fine-tuning are widely available.

Disadvantages:

a. **High Dependency:** The success of fine-tuning largely depends on the quality and relevance of the pre-trained model. If the base model's initial training was not comprehensive or relevant to the target domain, performance might be compromised.

b. **Potential for Overfitting:** The model performs well on training data but poorly on unseen data.

Given the constraints of our project——particularly the limited timeframe and the extensive resources required for full training, it was impractical to embark on building a model from scratch. Searching for and curating a dataset for full training would be a massive endeavor. **Furthermore,** the costs associated with the computational resources needed for developing a new model and deploying it on servers are exceedingly high. **Thus,** we opted to leverage the capabilities of an

existing pre-trained model. By fine-tuning, we could still meet the specific needs of our project without the substantial overheads of full training. This approach not only saved time and resources but also allowed us to make rapid iterations and improvements, which is crucial for the dynamic environment of our project.

This strategic decision enabled us to develop a robust solution efficiently and cost-effectively, so that we could deliver value quickly while maintaining high standards of model performance and reliability.

2) Baseline Comparison:

After deciding on the method for training our model, we needed to select a pre-trained LLM model that fits the specific needs of our project. The simplest and most direct method was to search for relevant models on **Hugging Face.** We limited our choices by considering the model's popularity (number of stars) and the size of the model parameters (less than 13B, which would occupy approximately 50GB of GPU resources using full precision FP32). We identified **META-LLaMA-2-7B and META-LLaMA-2-13B** as potential candidates. **Additionally,** considering the immense popularity and excellent response quality of the **GPT-3.5** model, it was reasonable to include **GPT-3.5** in our list of candidates.

Next, we conducted a detailed comparison of these three models based on the requirements for fine-tuning, and the output performance, and we ultimately chose **GPT-3.5-Turbo** as our pre-trained model.

META-LLaMA-2-7B

Advantages:

a. **Parameter Efficiency:** The **META-LLaMA-2-7B,** with 7 billion parameters, offers a balance between computational efficiency and capability. It requires less GPU memory compared to **META-LLaMa-2-13B.**

b. **Adaptability:** Being a smaller model, it can be more quickly adapted to specific tasks through fine-tuning. This makes it suitable for projects that need rapid development cycles and frequent updates.

c. **Lower Operating Costs:** The reduced size also implies lower operational costs in terms of power consumption and hardware requirements.

Disadvantages:

a. **Limited Understanding:** With fewer parameters, the **META-LLaMA-2-7B** may struggle with complex language understanding tasks.

b. **Limited Adaptation** of Various System: The **META-LlaMA** only supports **Linux OS** system which can be a challenge to deploy it in a server and make consistent connections between clients and server.

META-LLaMA-2-13B

Advantages:

a. **Increased Capability:** With 13 billion parameters, this model offers more enhanced understanding and generative capabilities compared to its smaller counterpart.
b. **Performance:** It strikes a good balance between size and performance, capable of handling more detailed and complex interactions than the 7B model.
c. **Flexibility:** It provides a good platform for both general and specialized tasks, making it a versatile choice for projects that might evolve over time.

Disadvantages:
Same as **META-LlaMA-2-7B**

GPT-3.5-Turbo
Advantages:

a. **State-of-the-Art Performance: GPT-3.5-Turbo** is known for its exceptional language understanding and generation capabilities, making it one of the top choices for applications requiring high-quality outputs.
b. **Broad Knowledge Base:** Its training on a diverse and extensive dataset allows it to perform well across a wide range of tasks without significant fine-tuning.
c. **Efficiency in Handling Complex Tasks:** The model is particularly adept at handling complex queries and generating detailed, nuanced responses, which is beneficial for advanced applications like content creation especially for plots generation.

For the specific operational constraints and requirements of our project, it becomes evident that two **META-LLaMa** models, presented significant limitations. **Firstly,** these models are supported exclusively on **Linux OS** systems, which restricts their applicability across different operating environments, thereby limiting their versatility for diverse project deployments. **Furthermore,** the substantial parameter size of both the 7B and 13B models necessitates at least one 48GB professional training GPU, such as the NVIDIA A6000, for training, fine-tuning, deployment, and inference processes. This not only increases the infrastructural costs but also elevates the operational expenses significantly.

Additionally, given that the 7B and 13B models serve as entry-level options within the **META-LLaMa** series, their actual inference performance significantly lags behind that of **GPT-3.5.** The **GPT-3.5** model, on the other hand, does not require local GPU resources for fine-tuning or inference, as it can be efficiently managed via an officially provided API interface. This aspect markedly reduces the overhead associated with the model's use.

Moreover, the cost-effectiveness of using **GPT-3.5** is particularly compelling. The fine-tuning costs are extremely low, amounting to merely $0.008 per 1,000 tokens, and the inference costs are even lower, at $0.003 per 1,000 tokens. These factors make **GPT-3.5** not only a technically superior choice but also a financially viable one, aligning seamlessly with both the performance expectations and budgetary constraints of our software project. **Consequently,**

GPT-3.5 stands out as the optimal choice for our needs, offering a blend of high efficiency, robust performance, and cost-effectiveness that is crucial for the success and scalability of our project.

3. Dataset preparing and preprocessing

Data collection was the first and most fundamental step of the project. It supplied the essential raw material for subsequent model training, fine-tuning, and testing. Without sufficient high-quality data, the model's performance and practical application would be significantly compromised. Before starting the collection process, we needed to settle on the data requirements.

1) Define Data requirement

In the development of our game, a critical initial step involved establishing the main storyline along with potential branching plots that enhance player engagement and decision-making. We meticulously defined various scenes and constructed character dialogues, integrating choices and interactions that players can undertake. This not only enriched the narrative but also allowed for a more interactive and personalized gaming experience.

A coherent and distinct language style and tone were essential components of our game's design. We determined the overall language style——whether formal, informal, or humorous——based on the game's theme and target audience. Consistency in language and tone across different characters was rigorously maintained to preserve the game's narrative integrity and to enhance the player's immersion. Each character was crafted with a unique voice that reflects their personality and background, contributing to a more dynamic and believable world.

2) Data collection

For the manual data writing phase of our project, we focused on creating a diverse range of plot and dialogue samples. This was done to adequately cover all possible branching scenarios in the game, ensuring a rich narrative experience for players. **Additionally,** we collected texts that exhibited a variety of language styles and tones. This collection served as training data, which was crucial for developing a nuanced and engaging dialogue system that can adapt to different narrative contexts and player choices.

To complement our manual efforts, we also implemented an automated data generation strategy. This involved leveraging existing resources such as text-based adventure games, novels, and dialogue scripts to mine data relevant to our game's narrative needs. **Furthermore,** we utilized script tools to scrape dialogues and story plots from the internet, efficiently gathering a large and diverse dataset. This approach not only expedited the data collection process but also introduced a broader spectrum of narrative elements and stylistic variations to enrich our game’s content.

4. Full-Stack Website Implementation

One of the core objectives of this project was to develop a web-based UI design that is highly interactive and visually appealing, encompassing two main

pages: the homepage (index.html) and the chat interface (chat.html). To achieve this, we opted for **Bootstrap** and **Flask** frameworks as our foundational technologies, incorporating **OpenAI's API** to achieve the goal of interacting with OpenAI to create new stories.

Prompt engineering was also implement in this part, to enhance the performance of our own modal, and applied the application into an interactive storytelling game with state management and dynamic responses based on user choices.

1) Front-End

a. Bootstrap Framework

Bootstrap is a widely-used front-end framework that offers a responsive grid system and a variety of ready-made components, which significantly accelerates the front-end development process. In this project, we utilized the Bootstrap framework through the following steps:

i. **Layout Design:** Using **Bootstrap's** grid system, we designed layouts that adhere to UI design principles, ensuring optimal display across different devices. The system's flexibility supports responsive features like column wrapping and offsetting, enhancing accessibility on devices from desktops to mobiles.

ii. **Component Selection and Customization:** We selected essential **Bootstrap** components——navigation bars, forms, and buttons——and customize them with **CSS** to align with the design and brand image. This included modifying shapes, sizes, and effects to create a unique, functional user interface.

iii. **Enhanced Interactivity:** Interactivity was boosted using **Bootstrap's JavaScript plugins,** such as modal windows for in-page content and dropdown menus for compact navigation. Tools like tooltips and collapsible elements were also integrated, improving the dynamic user experience and site usability.

Through these detailed and thoughtful implementations of **Bootstrap's** capabilities, we ensured that the website is not only functional and responsive but also engaging and representative of the brand's image, providing a solid foundation for user interaction and satisfaction.

b. JavaScript and AJAX

To enhance the interactive experience for users, the project employed JavaScript and AJAX for asynchronous data interaction on the client side:

AJAX Calls: On the chat.html page, **AJAX** technology was used to submit form data asynchronously, allowing users to send messages and receive responses from OpenAI without reloading the page.

Frontend Logic Handling: JavaScript was used to preprocess user inputs (such as validation and formatting) and dynamically update the chat interface upon receiving API responses.

2) Back-End

a. Flash Framework

Flask is a lightweight Python web framework renowned for its simplicity and flexibility, which are key attributes that facilitate rapid development and prototyping. Unlike more heavyweight frameworks that come with a default set of tools and libraries, **Flask** provides developers with the bare essential, enabling them to add only the tools and libraries they need. This "micro-framework" approach makes **Flask** particularly suitable for small to medium applications and for developers who prefer fine-grained control over their application components:

i. **Server Setup: Flask** acts as the backbone of the server-side logic, handling HTTP requests and responses. Its built-in development server is used during the development phase for testing and debugging purposes, offering a straightforward setup for running the application locally.

ii. **Routing Configuration: Flask's** URL routing mechanism is used to map URLs to specific functions in the code. This is essential for a web application where different pages and functionalities are accessed via distinct URLs. **Flask's** routing provides a way to capture parameters from URLs, which can be used to generate dynamic content based on user actions or preferences.

iii. **Integration with OpenAI API:** In the context of this project, **Flask** handles interactions with the OpenAI API. It manages the requests sent to the API and processes the responses that are received. Flask routes these requests from the web pages, ensuring that data fetched from OpenAI is correctly displayed to the users or utilized in the application logic.

By leveraging **Flask's** capabilities, this project could maintain a lightweight structure while providing robust and efficient back-end functionalities, which is essential for handling complex operations such as API interactions and dynamic web page rendering.

b. Logic of back-end

i. **State Management:** Each session of game should manage a range of variables that manage the state of the games, including the current motif, the player's progress, and game outcomes (victory/defeat). Variables such as motif, initial_prompt, option_dict, victory, and defeat are stored in the session to maintain continuity across multiple interactions.

ii. **Dynamic Prompt Generation:** In the back end, the input of user is preprocessed by prompt engineering to facilitate the generation of story plot. When the game starts, the initialize_game function sets the initial scenario based on the user's choice of motif. **Next,** options are also parsed from the response and sent the corresponding choices to the LLM by create_option_dict function.

iii. **Custom Response Handling:** To offer a clear instruction, we implemented the website to provide clear instructions and handles invalid inputs, enhancing user experience. **Meanwhile,** functions such as

check_game_outcome analyze the response to determine if the player has won, lost or continues the game.

iv. **Error Handling and Debugging:** In our back end, we employed the comprehensive error handling to ensure the application gracefully handles issues and provides meaningful error messages. Print statements are used to debug and trace the application's state and interactions.

5. Testing and evaluation

Testing strategy plays a critical role in software development, ensuring the quality and reliability of software products while enhancing user satisfaction and reducing maintenance costs. During the project, we had adopted several testing methods to verify and validate the development processes, in the purpose of quality assurance and satisfying the requirements of users.

For the sake of software quality assurance, our team implemented verification and validation processes concurrently with the development phases, which would reduce the development cycle time, allowing early detection of defects and bugs.

We employed white box test for the full-stack website development. In detail, two of our team members performed code review of each other's code, including internal structures and workings of the application. This method was particularly effective in checking for logical errors in our prompt engineer process and correctness of data preparation. **What is more,** we encouraged the good code style and abandon unclear structure of the code, reducing the burden of maintenance later.

Moreover, unit testing was the main testing for the functionality, which would entail creating multiple test cases and covering both positive, negative and edge cases. For the UI, we performed a series of test for the elements like forms, navigation, and user interactions. For backend, we focused on testing API connections and prompt engineer function to ensure logic consistency. **For instance,** in our game, user should input the number 1, 2, 3 as the selection of behavior. To examine the effectiveness of detecting and reporting invalid input, we selected several test inputs to cover as much as kind of input to our chat box. In addition to that, the mechanism to detect the end of the game was one of the major tests. It's worth mentioned that, we achieved keeping track of the status of the game through the conversation and deciding when to end the game successfully.

Integration testing was performed after the frontend and backend are coherent. In this test phase, we verified that different modules work together seamlessly. It is a crucial stage that it could catching issues that occur when individual pieces of the application interact, ensuring that the **integr4ation** of components does not lead to functionality failures.

In our AI-driven project, the performance of AI model is another aspect that needs to be tested and evaluated. As we were taking the fine-tuning model as our final choose, we would particularly investigate whether the model after fine tuned align with our objectives. In this case, **Mean Squared Error** could be a good evaluation metrics, and other more relevant metrics are assessed as well.

In the final phase of our project, we conducted user acceptance testing through two distinct methods, each focusing on specific aspects of user interaction and interface design.

Firstly, we implemented **A/B testing** to finalize our user interface design. During this phase, two versions of our application were compared head-to-head to determine which one was more favored by the users. We collected specific data during these tests and used a **t-test** to statistically analyze the significant differences between the two website designs. This method helped us make informed decisions based on user preferences and interactions.

Secondly, we conducted a beta testing phase, during which we gathered comprehensive user feedback for later analysis and evaluation. We identified six key categories of feedback for collection. Three of these categories were automatically collected and used to analyze the game's acceptance, specifically by comparing the number of visitors to the webpage with those who completed a game session. The other three categories involved subjective evaluations by users, focusing on their overall rating of the game, the continuity of the game's storyline, and the game's difficulty level.

These attributes were carefully chosen to align with our game's objectives. By analyzing the game ratings, we could gauge the enjoyment users derived from the game. Assessing the storyline continuity allowed us to evaluate both the performance of our model and the game's ability to engage and immerse players.

Lastly, by examining the game's difficulty, we aimed to enhance the players' problem-solving and critical thinking skills, making the game not only entertaining but also intellectually stimulating.

V. Development Process

1. **Dataset & Model:**
   - i. **Planning Phase**

     Background Selection:

     - **Research and Discussion:** We conducted extensive research and discussions to determine the most popular text adventure game backgrounds in the market. We finalized three themes: **Island Survival, Post-Apocalyptic Zombies, and Magical World.**
     - **Reasoning:** These themes were based on their popularity in existing literature, games, and media, ensuring broad appeal and a wealth of reference material.

   - ii. **Design Phase**

     a. **Data Format Specification:**

     - **Structure Definition:** According to OpenAI's official documentation, the first step in model fine-tuning was preparing a dataset for the downstream task. In alignment with the guidelines detailed in OpenAI's official documentation for model fine-tuning, our approach of constructing the dataset involved crafting each example to simulate realistic interactions, much like those anticipated in production environments. The structure of

each conversation within the dataset mirrors that of our **Chat Completions API.** This entailed organizing the data into a list of messages, with each message assigned a specific role (‘system’, ‘user’, or ‘assistant’). Each message contains relevant content, and where necessary, an optional name is provided. It was important for the integrity and utility of the training process that our dataset not only replicate typical user interactions but also address specific instances where previous model outputs did not meet our expectations. **To this end,** some examples were deliberately designed to include these less-than-ideal scenarios to ensure that the training directly enhances the model's performance in these critical areas. The assistant's responses were carefully crafted to reflect the optimal outcomes we aspire for the model to replicate, thereby serving as a benchmark for desired behavior. The dataset also accommodates multiple messages within a single example that assume the assistant role, reflecting the complex and varied nature of real-world interactions. During the fine-tuning process, the standard protocol is to engage with all assistant messages present in each example.

We will employ JSONL files to store our dataset, where each JSON object represents a complete conversation.

b. Source Selection:

- **Literature Source:** We had manually selected a range of novels, movies, TV shows, games, articles on blogs and websites, movie scripts and dialogues related to the three themes to provide a rich source of diverse dialogues, scenes, and plot lines.
- **Tools and Methods:** We decided to use **GPT-4** to generate data automatically.
  - **Validation:** Conduct preliminary analysis of the chosen literature and sources to ensure their content is relevant to the themes and provides sufficient information.
  - **Verification:** Confirm the selected tools and methods with team members and conduct small-scale tests to ensure their effectiveness.

c. Interactivity and Choice Design:

- **Story Branches:** Designed multiple story paths and interaction points to ensure high interactivity. Created branching story structures with various outcomes for each theme.
- **Dialogue Diversity:** Designed diverse dialogue options for each interaction to ensure that different player choices lead to different outcomes.
  - **Validation:** Validate the reasonableness and appeal of interactivity and choice design.
  - **Verification:** Discuss with the design team and user experience experts to ensure the interactivity and choice design align with project goals and player expectations.

iii. Implementation Phase

a. Data Collection:

- **Manual Data Creation:** Initially, we downloaded the literature phase of the novels, movies, TV shows, games, articles on blogs and websites, movie scripts and dialogues on the internet, we had also used **Scrapy** to help us grasp the scripts and dialogues from the internet. Then a group of writers read the selected literature and manually created data samples according to the defined format. Use one data as a example., we first set the fundamental emotion and condition of this data, the 'role' which is 'system' contains those data, then according to the literature, we split the content into different parts which belongs to different roles like 'user', 'assistant' and so on.
- **Automated Data Generation:** Utilized **GPT-4** to generate more data. Input the format of our data to the GPT for it to learn how to generate such data in the right format. Then input the keywords and directory into GPT to generate coherent dialogues, scenes, and plot.
  - **Validation:** Perform quality checks on the generated data to ensure it meets the data format and annotation guidelines.
  - **Verification:** Regularly review the data collection progress and quality to ensure it is completed as planned and meets the expected standards.

b. Data Checking:

In this phase, we rigorously ensured that the story plot remains free of inappropriate language, such as slurs or derogatory terms, to maintain a suitable environment for educational purposes. **Additionally,** we strategically designed some plot branches to result in adverse outcomes, enhancing the game's challenge and encouraging players to engage in critical thinking and problem-solving. **Lastly,** to enable the model to generate plausible and actionable choices for the player, we had carefully defined the "user" variables to include a range of movements and actions. This approach helps the model recognize and replicate this pattern, ensuring that the game interactions remain dynamic and contextually appropriate **Interactivity and Choice Integration:**

- **Interactive Paths:** Designed and documented multiple story paths and outcomes. Annotated each data entry with possible choices and resulting branches.
- **Dialogue Design:** Created diverse dialogue options for each scene to ensure different choices lead to different story developments.

c. Verification and Validation:

- **Data Quality Checks:** Conducted regular quality checks to ensure data accuracy and relevance. Used automated scripts and manual reviews to validate the integrity and consistency of the dataset.
- **Functionality Testing:** Integrated the data into the model and tested its performance in various scenarios. Ensured that the interactive elements functioned as intended and made adjustments based on feedback.

Tools Used:

- **GPT-4:** For data generation, noise reduction, and data enhancement.

iv. Fine-tuning Model

Upon successfully uploading our dataset file to the OpenAI platform, we proceeded with creating the fine-tuning task using the following Python script:

```
from openai import OpenAI
client = OpenAI(
    api_key="")

response = client.fine_tuning.jobs.create(
training_file="file-BgXZSK8etUQrtSAGKPBZ6XY3",
validation_file="file-BgXdsDfDFw3e273HDWsjkdhD",
    model="gpt-3.5-turbo",
    suffix="AI-Dungeon",
    hyperparameters={
        "n_epochs": 3
    }
)

modelinfo = open("modelinfo.txt", "w")
print(response, file=modelinfo)
```

We recorded the unique identifier returned in the response from the file upload, ensuring we keep track of our dataset's location on the platform. **Furthermore,** we have selected the pre-trained model **GPT-3.5 Turbo** for our fine-tuning purposes.

Subsequently, we configured the hyper-parameters for our model. We plan to set the number of training epochs to 3, as recommended by OpenAI, which is generally sufficient to achieve notable improvements in model performance.

Additionally, we split our dataset into a training set and a validation set. This division will enable us to assess the model's performance comprehensively during and after the fine-tuning process, ensuring that our model not only learns effectively but also generalizes well to new, unseen data. This structured approach guarantees that our fine-tuning task is set up for success, optimizing our model's ability to deliver high-quality, reliable results in production scenarios.

Finally, if we successfully create our fine-tuned model, OpenAI would return the model specific information about this model with unique model id, we could use the model though OpenAI library to deploy it into our website with router.

2. **Full Stack Website:**
   i. **Planning Phase**

In this initial phase, we undertook a comprehensive evaluation to define the project's scope and objectives. Our primary focus was to develop a responsive web interface comprising two main pages: index.html for the homepage and chat.html for the chat interface. A key objective was to incorporate OpenAI's API into chat.html to facilitate intelligent user interactions, such as automated responses and interactive dialogues. We selected **Bootstrap** to ensure rapid prototyping and to maintain design responsiveness across various devices. **Flask** was chosen for its lightweight structure and ease of integration with backend APIs, making it ideal

for our server-side requirements. This phase also involved an in-depth selection process for the right tools, frameworks, and architectural approaches that would align best with the project's needs, emphasizing scalability and maintainability.

ii. Design Phase

During the design stage, we leveraged **Bootstrap** to develop wireframes and mockups, which served as visual guides for the layout and user flow of the website. Special attention was dedicated to **user experience (UX) design,** ensuring that the interface was intuitive and user-friendly on various devices, from desktops to mobile phones. We meticulously designed visual elements such as color schemes, typography, and component aesthetics to ensure they were cohesive and reflective of the brand's identity. This phase was crucial for setting the visual direction and ensuring that all graphical elements functioned harmoniously across different viewing platforms.

iii. Implementation Phase

The implementation phase was divided into two critical components: frontend development with **Bootstrap** and backend setup with **Flask.** Using **Bootstrap,** we rapidly structured the pages and customized UI components for both index.html and chat.html, ensuring that they were visually appealing and functionally robust. **Concurrently, Flask** was configured to manage the server-side logic, including establishing routes to handle requests and responses for OpenAI's API integrations. We employed **JavaScript and AJAX** to facilitate seamless API communication, enabling dynamic data exchange and real-time content updates without the need to reload the web pages.

iv. Testing Phase

Our testing phase was thorough, involving multiple layers of tests, including **unit testing** to ensure individual components functioned correctly, and **user acceptance testing (UAT)** to gauge the application's performance from an end-user perspective. We conducted tests across a variety of devices to verify the integrity of the responsive design. The **Flask** backend and the API integration underwent stringent testing to handle potential errors and to ensure stable and reliable data exchanges. We addressed challenges like asynchronous request management and API rate limiting by implementing advanced error handling strategies and optimizing request processing to enhance overall system resilience and performance.

v. Deployment Phase

Deployment involved transferring the fully tested application to a production environment. This phase required meticulous planning to ensure the server and hosting environment were secure and optimized for performance, focusing on load speeds and response times. We established robust **continuous integration/ continuous deployment (CI/CD)** pipelines to facilitate smooth future updates and maintenance, allowing for streamlined development cycles and quicker release schedules.

vi. Challenges and Solutions

We encountered significant challenges, particularly with the integration of the OpenAI API. Ensuring efficient and secure API communication was paramount; we addressed this by developing secure backend services that could safely handle API requests while protecting sensitive data like API keys. Another major challenge was maintaining consistent responsiveness across all devices, especially when integrating complex **JavaScript** interactions. Through iterative testing and continuous refinements of **Bootstrap** configurations, we achieved a highly responsive and consistent user experience.

VI. Result & Analysis

1. Website:

a. Compatibility Test:

Our web support different browsers including **Google Chrome, Firefox and Edge.** It also supports **Windows, MacOS and Android.** We have designed the grid system flexibly to adapt different size screens and windows. **However,** when we put our web on a mobile phone, it occurs incompatible problems, such as not applicable for the grid system.

b. Load Capacity Test:

Our website is designed to support up to 30 concurrent users at its peak capacity. **For optimal performance,** the safe threshold for concurrent users is between 15 to 20. Within this range, the website maintains smooth functionality and avoids significant lag or delays. When operating within this optimal range, the system can efficiently process between 5 to 7 requests per second, ensuring a responsive and reliable user experience.

c. Response Time:

The average time for our website's first screen to load is approximately 2.3 seconds at its longest. The average time for the entire page to load completely is about 4.1 seconds. The page interaction time, which is the time it takes for the website to respond and load after a user clicks an element, is roughly 1.4 seconds on average. **Additionally,** when a user inputs text, the average time for the backend AI to generate and display a response on the page is around 2.3 seconds.

d. A/B Testing:

We conducted **A/B testing** regarding the layout of the web page. For example, on the index.html page, both the "Play Now" button in the navigation bar and the "Start Playing" button in the center of the page redirect to the chat.html page. We investigated the difference in the number of users who logged into the site and chose to go to chat.html with and without the "Play Now" button. We found that having two entry buttons led to a slight increase in the number of users entering chat.html, indicating that providing more entry points can increase the likelihood of users engaging with our game.

2. **User Feedback:**
   a. **Analytical Approach**

Genuine feedback obtained from players is pivotal for the analysis and development of games. **Therefore,** we utilize a series of metrics to thoroughly analyze this feedback. One such metric is the **Complete Gameplay Instances,** which is calculated using the formula:

Complete Gameplay Instances = Total Website Visits * Conversion Rate to Start Playing * Conversion Rate to Complete Playing

This metric is crucial as it quantifies how many visitors to our website engage deeply with the game by playing it through to completion.

To enhance our understanding of the players' experiences, we collect data through offline surveys and have derived three analytical metrics from the complete gameplay instances:

a) **Game Approval Rate:** This represents the percentage of players who have positively rated the game, providing insight into the game's overall reception.

b) **Continuity Evaluation Rate of the Game Storyline:** This measures how players perceive the coherence and flow of the game's narrative, an essential aspect of storytelling in games.

c) **Game Difficulty Evaluation Rate:** This assesses how players rate the challenge level of the game, indicating whether the difficulty is well-balanced.

For each of these feedback metrics, we conduct an independent funnel analysis. **Funnel analysis** is a powerful tool that helps us to visualize the conversion rates at various stages of the player's engagement with the game. By quantitatively analyzing these funnels, we can effectively measure and understand our feedback data, allowing us to conduct a comprehensive analysis of the game. This method not only highlights areas of success but also pinpoints aspects that may require further refinement to enhance the player's experience and satisfaction.

   b. **Funnel Analysis:**

Based on the data provided, we conducted three independent funnel analyses, each targeting different user feedback metrics: **approval rate, continuity evaluation rate of the game storyline, and game difficulty evaluation rate.** Here is a detailed analysis and conclusions for each funnel:

Basic Data

- Total number of webpage opens: 217 visits

- Conversion rate to start playing: 88%

- Conversion rate to complete playing: 56%

Funnel Analysis 1: Approval Rate

1. Number of people who started the game:

- 217 * 88% = 191 people

2. Number of people who completed the game:

- 191 * 56% = 107 people

3. Number of people who gave a positive review:

- 107 * 78% = 83 people

Funnel Analysis 2: Game Storyline Continuity Evaluation Rate

1. Number of people who started the game:

- 191 people

2. Number of people who completed the game:

- 107 people

3. Number of people who rated the storyline positively:

- 107 * 62% = 66 people

Funnel Analysis 3: Game Difficulty Evaluation Rate

1. Number of people who started the game:

- 191 people

2. Number of people who completed the game:

- 107 people

3. Number of people who rated the game difficulty positively:

- 107 * 42% = 45 people

a. Conclusions and Recommendations

i. Approval Rate Analysis:

The approval rate stands at **78%,** indicating that the vast majority of players who completed the game had a positive experience. This is a highly encouraging metric, suggesting that the game successfully meets player expectations.

ii. Game Storyline Continuity Evaluation:

The continuity evaluation rate of the game storyline is **62%.** Although the majority of users found the storyline to be coherent, there is room for improvement. Enhancements could be made to make the narrative more engaging and seamless.

iii. Game Difficulty Evaluation:

The evaluation rate for game difficulty is the lowest at **42%,** suggesting that many players find the game too challenging or unbalanced. It is recommended to adjust the difficulty settings based on player feedback. If feasible, introducing multiple difficulty levels could accommodate players of varying skill levels.

Overall, the game performs well in engaging players but requires further optimization in storyline continuity and difficulty balancing. Improvements in these areas could significantly enhance overall player satisfaction and market performance of the game.

This structured analysis provided a clear pathway for refining game elements to better meet the needs of our players, ensuring an enjoyable and challenging experience for a diverse gaming community.

7. Discussion

Throughout the development, we encountered many challenges and learnt techniques to address them:

1. **Resource Capacity:** For example, when we try to use **META-LLaMA-2-7B,** we found that the available resource couldn't load the model due to a lack of memory. Even though downgrade the precision should ease the problem, but the performance was terribly bad. To avoid this obstacle, we used OpenAI's model which provides an API to run online with capable server.
2. **GPT data generation:** It could be a huge burden if we chose to collect all the data manually. Thus, we used **GPT-4** to generate text given formatted examples, expanding the size of the dataset while maintaining the format, greatly reduced the time cost.
3. **Model Performance:** We also found that the GPT model's responses were too positive, making the game less interesting. To address this issue we did some research on prompt engineering and emphasized the negative responses, thus the model could generate bad endings more often than before.
4. **Future Improvements:**
   i. **Gather more data for fine-tuning**
   ii. **Optimize the logic of backend**
   iii. **Design a better UI**
   iv. **And more…**

8. Conclusion

The project to develop an AI-driven text adventure game was successful in engaging players and providing a novel way to address mental health support. Despite some challenges and areas needing improvement, the overall impact of the project has been positive, highlighting its potential in the market.

Key Takeaways:

1. **Engagement:** The game was well-received, with a **78%** approval rate among those who completed it, indicating strong engagement and satisfaction.
2. **Storyline and Difficulty:** While the majority of players appreciated the storyline, with a **62%** positive evaluation rate, the game's difficulty level received lower ratings at **42%.** This suggests a need for balancing the game's difficulty to cater to a broader audience.
3. **Technical Challenges:** Resource limitations led to a switch from the initially chosen **META-LLaMA-2** model to OpenAI's **GPT-3.5 Turbo,** which proved effective. Data generation and prompt engineering were crucial in addressing issues with overly positive model responses.

Impact: The project has demonstrated that AI-driven text adventure games can be a viable tool for mental health support and stress relief. The positive feedback indicates that such games can effectively engage users and provide them with an enjoyable experience.

Future Work: To enhance the project, the following steps are recommended:

1. **Data Collection:** Gather more diverse and comprehensive data for further fine-tuning of the model.
2. **Backend Optimization:** Improve backend logic to handle user inputs more efficiently.
3. **UI Enhancement:** Redesign the user interface to be more intuitive and visually appealing.
4. **Difficulty Levels:** Introduce multiple difficulty levels to accommodate players with varying skills and preferences.

These improvements will help refine the game's elements, ensuring it meets the needs of a diverse gaming community and continues to provide a valuable tool for mental health support.