ECE495 Final Project

Jessica Chen

December 13, 2024

Due to the project nature, I decided to make a github for this project with all the files. The only things not added to the github are of course any API keys, as well as the scraped wiki pages as that would be a lot: https://github.com/jess-che/terraria-rag

1 Project Overview

Terraria is a 2D sandbox game that emphasizes exploration, crafting, and combat. While it is often compared to other sandbox games like Minecraft, one of Terraria's defining features is its vast array of monsters, boss fights, items, and weapons. This immense quantity of content sets it apart from other games in the genre. However, this depth and complexity also present a challenge, as even experienced Terraria players frequently turn to external resources, such as the Terraria Wiki, to find specific information during gameplay.

Although Terraria includes an in-game guide to assist players with crafting recipes and general gameplay tips, the guidance it provides is limited. The tips offered are triggered by how frequently a player interacts with the guide, which can result in information that may not always be relevant to the player's current needs.

Thus, the goal of this project is to develop a more interactive guide for Terraria using a Retrieval-Augmented Generation (RAG) system. This system will combine the retrieval capabilities of a BERT-based model with the generative abilities of a GPT-based model to create a more dynamic and personalized guide to terraria compared to looking at the wiki pages or trying to learn from the in game guide.

2 Specifications

Given the vast scope of Terraria, which features thousands of potential queries, and the comprehensive nature of the Terraria Wiki — often including details beyond the typical player's interest (such as historical patch notes and obscure mechanics) — this chatbot will prioritize addressing questions relevant to a standard gameplay experience. It will focus on providing clear, practical guidance that aligns with the typical needs and questions of an average player during normal playthroughs, specifically:

- **Crafting Items:** Queries related to crafting specific items (e.g., *How do I craft the Terra Blade?*).
- **Obtaining Items:** Questions about where and how to acquire particular items (e.g., *Where can I find Chlorophyte Ore?*).
- **General Tips and Information:** Helpful advice or general gameplay queries especially in regards on how to progress through bosses (e.g., *How can I summon the Eye of Cthulhu?*).

3 Data Collection and Preprocessing

The first step in the process was to scrape and clean the data from the Terraria Wiki.

- Using the Terraria Wiki API, I developed a web scraper to retrieve all pages from the wiki. Each page's content was downloaded as an HTML document using the script *scraper.py*.

The next step involved removing redundant pages and those that did not fit the specifications for the bot.

- The first two categories of pages that were removed included redirect pages and pages that served as translations into different forms of English. These were programmatically filtered using the *remove_redundant_pages.py* script.
- Additional page removals were handled manually to ensure important information was not excluded. Below are the main categories of pages that were removed:
  - **Internal ID Pages:** Pages like "Accessory IDs" that primarily consist of graphical content and internal IDs. This information would be difficult to meaningfully convert into chatbot responses.
  - **Version/Platform Pages:** Pages such as "1.4.0.1" were excluded because they focused on version-specific changes rather than general gameplay. Platform-specific pages and platform histories were also removed. This aligns with the decision to focus on questions and answers that are relevant to players during a typical gameplay loop. More details on this can be found in the Past Iterations section.
  - **Out-of-Game Pages:** Pages that reference community content outside the game, such as information about the modded community or details about Relogic (the game developer), were also removed as they did not align with the chatbot's objectives.

After filtering the pages, I moved on to preprocessing the HTML data to be converted into JSON chunks. These chunks are used to improve the RAG system's retrieval capabilities. This preprocessing step underwent several iterations as I learned how the chunking affected model performance, how BERT responded to different formats, and how the Terraria Wiki's structure influenced chunking decisions.

- **Insight 1:** Metadata for each chunk should include both the page title and the section label. This allows the BERT model to prioritize matches not just by textual similarity but also by context.
- **Insight 2:** Since the Terraria Wiki follows certain structural patterns, I wrote specialized scripts to parse key sections accurately. This information in the structures tend to be information that corresponds to the main objective of the chatbot: general information, crafting details, and obtaining details. For less structured or unique sections, I implemented a general-purpose script to extract the content. This ensured that all relevant information was captured while maintaining consistency across different formats.

The following principles guided the preprocessing of the HTML data into JSON chunks:

1. **Logical Subdivision:** Each chunk was subdivided into the shortest logical component. For example, table data was processed so that each row became its own chunk. For paragraphs and bullet lists, each bullet point was treated as a separate chunk unless it was part of a nested list that required combined context for clarity. By focusing the content, BERT is less likely to retrieve content that contains certain words but is not connected to the overall question as well as ensure that when we truncate the chunks to stay within token confines, important information will not be cut out.
2. **Avoid Over-Segmentation:** Simple lists of items were not processed into chunks since either the chunk would be a giant list or there would be a separate chunk for each item, both ineffective retrieval. Furthermore, important information from these lists is typically available in other sections of the wiki.
3. **Modular Processing:** Each type of section was processed separately before being combined into a final JSON file. This allowed for greater control over the chunking process and made it easier to fine-tune how each section was parsed. For example, sections like *Variants* had varied underlying HTML structures, so dedicated scripts were developed to handle the nuances of each section type. This modular approach allowed for consistent formatting and ensured readability in the final JSON output.

(a) While this step constituted a large amount to improving the results of the RAG system and was time consuming, especially since sections like *Variants* had inconsistent HTML structures, I will not go into too much detail about how each section was processed as the main focus on this report is on the RAG system. However, information in past iterations does give insights on how I arrived to the current preprocessing system based on the performance of the RAG system in past information.

4 RAG System

Since the project called for a BERT-based model, I initially selected the `all-MiniLM-L6-v2` model. This model maps sentences and paragraphs to a dense vector space and is optimized for short sentences (up to 256 tokens). Given that each input text for the ChatGPT prompt was limited to 200 tokens, this model's token limitation was sufficient for the task.
My decision to start with `all-MiniLM-L6-v2` was informed by the Sentence Transformer's documentation, which highlights the model's balance of computational efficiency and embedding quality for general-purpose tasks. Since my goal was to iterate quickly and refine the model over multiple development cycles, `all-MiniLM-L6-v2` aligned well with this approach due to its relatively fast training time and acceptable embedding quality.
After achieving initial satisfactory preprocessing, I also experimented with the `bert-uncased` model from Hugging Face. However, I observed that the embedding step using `bert-uncased` was significantly more time-consuming. Given this, I opted to continue using the `all-MiniLM-L6-v2` model, as I continued to improve the preprocessing.
In the screenshot below, we can see the inital model behavior with just using the base retrieval system (comparing embeddings) with the top 5 most likely matches.

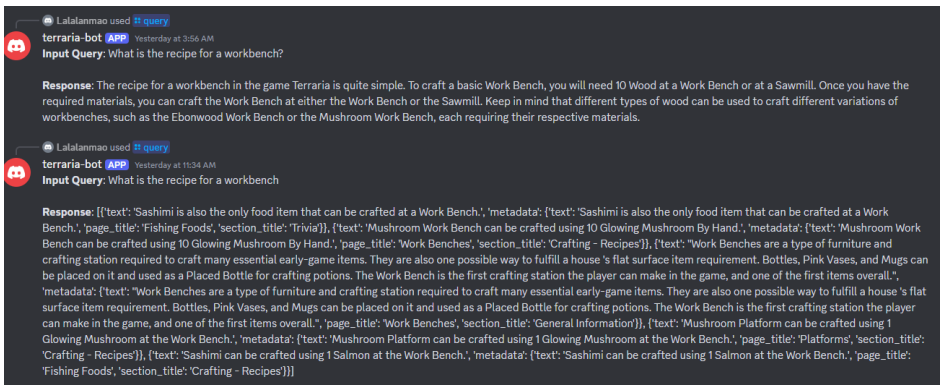


Figure 1: Pre BERT Score Changes

From this analysis, it became evident that a key flaw was the frequent occurrence of terms like "crafting" and "workbench" across multiple pages, not just within the specific section dedicated to crafting recipes. To address this, I incorporated an additional weighting mechanism for the score of the text. This weighting was based on the fuzzy matching of words from the input query against the page title and section title metadata.
The scoring system was adjusted to give a higher weight to page titles compared to section titles, as identifying the correct page is crucial for retrieving accurate information from the Terraria wiki rather than brief mentions in other pages. This change improved the relevance of the responses, as evidenced by higher alignment between the top results and user expectations.
The impact of this adjustment is shown in the results, where the most relevant information consistently receives a higher score. Moreover, after testing with various queries, I observed that the top 3 documents were typically the most relevant, while results beyond this threshold tended to be less useful.
Consequently, I refined the generation process to only utilize the top 3 most relevant documents for final output, thereby improving the efficiency and precision of the system.

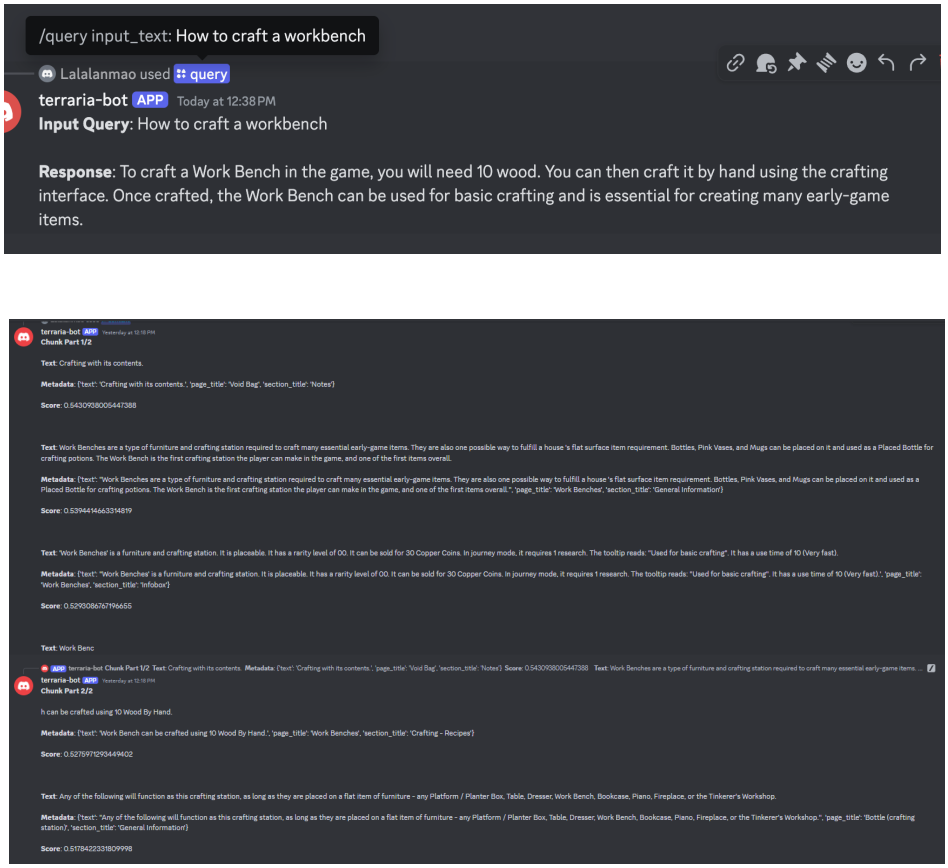


Figure 2: Post BERT Score Changes

After I was satisfied with the results of retrieval, the next step was to modify the prompt generation for the model. Currently my initial testing prompt was a generic: "You are a helpful assistant that provides detailed and accurate responses."
From reading a few discussions such as the one found here, the two key tips are to avoid negative instructions as it favors positive instructions more, and that GPT-3 models, which is the type of model I was using for generation tends to prefer information toward the end of the prompt.
Using this information, I made several modifications to the way input queries were formatted. I began by providing initial context about the purpose and goals of the generation task, as well as specifying how the information would be used. This context was reinforced at both the start and end of the prompt to ensure alignment throughout the process.
Since the bot is specifically designed for Terraria-related queries, I included the following directive within the prompt: *"Provide step-by-step concise instructions, specific item names, and game-related terminology when relevant."* This ensured that the bot maintained a clear focus on Terraria-specific content, with step-by-step guidance promoting structured, easy-to-follow responses. While this approach led to many answers being formatted as lists, I think overall it improved the quality of the generated responses.

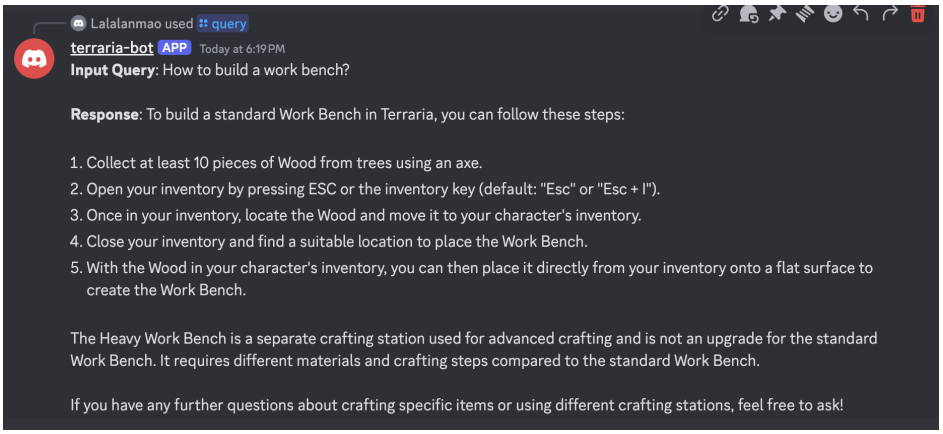


Figure 3: Post Prompt Generation Changes

Finally, I added an explicit instruction in the prompt to address situations where the provided context did not answer the question, as due to the nature of how the BERT was set up, even if all scores were low, it still returns documents. This prompts the bot to state clearly when it did not have the necessary information, reducing the likelihood of irrelevant or speculative responses if the question is unrelated to Terraria.

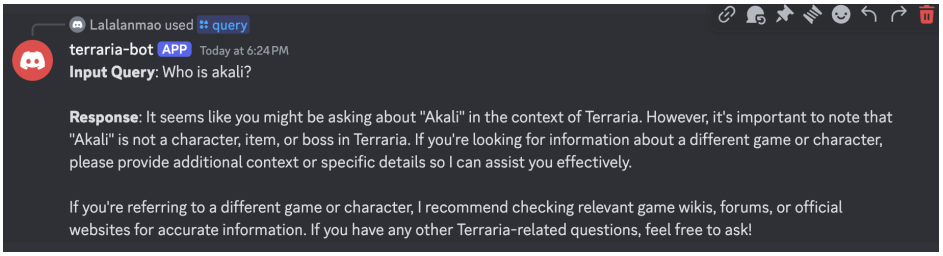


Figure 4: Prompt Unrelated to Terraria

5 Discord Integration

In order for the RAG system to actually function as a chat bot rather than something I feed information to in an array, I additionally added a discord feature into my RAG system.

In terms of RAG system and how it was run, with this discord integration:

- The conversion of metadata to the FAISS index was done in colab. (*index.py*).
- The index as well as the metadata json were then imported to the github environment so it could be accessed by the retrieval/generation in the discord bot. (*discord_bot.py*)

The discord bot could be interacted with using too main commands /query and /context.

- /query functions as the chatbot. The user provides its question in input text. This triggers the RAG system, and the discord bot sends the resulting generation in chat.

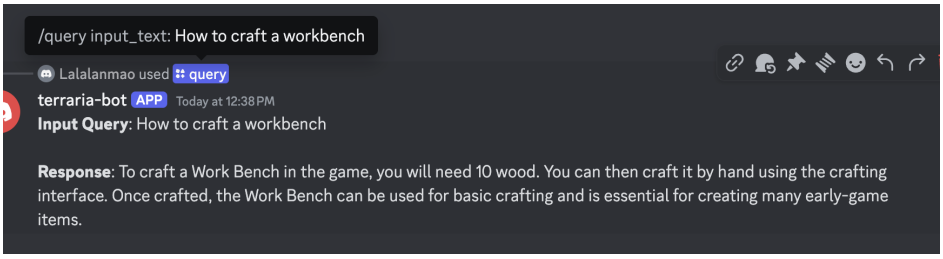


Figure 5: Example Query

- /context served more as a way to help me debug and improve the bert retrieval system of the model as it runs only the retrieval part of the RAG system and replies with the retrieved chunks as well as their corresponding score.

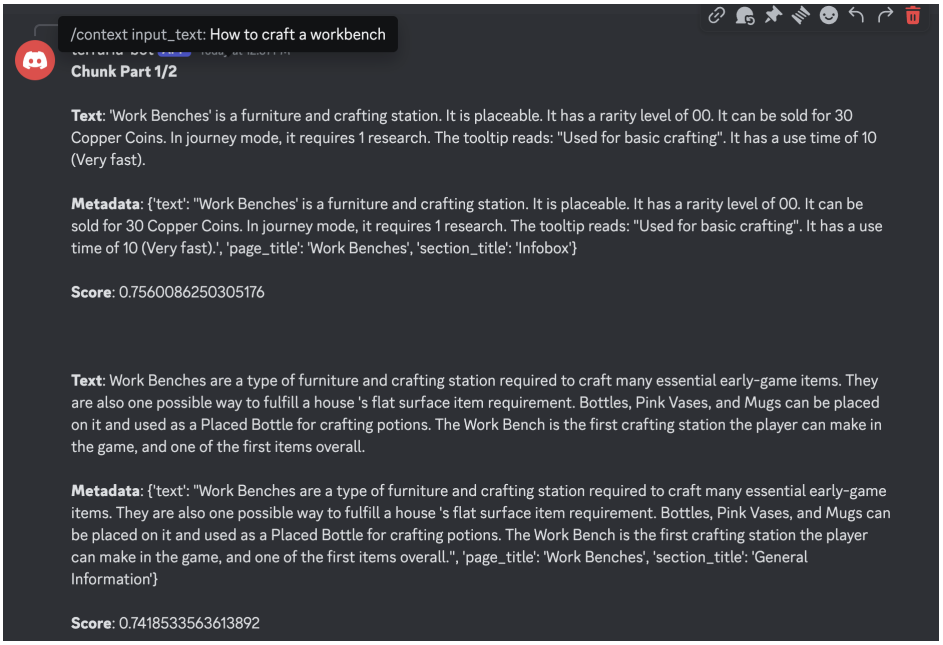


Figure 6: Example Context

This discord integration allows the chat to feel like it can actually be used and is part of a system as well as let other people other than myself interact with it (as you can see by the other username in the results section).

6 Results

Here are a few sample of results to show off the performance of the chatbot.

Crafting Questions

Obtaining Questions

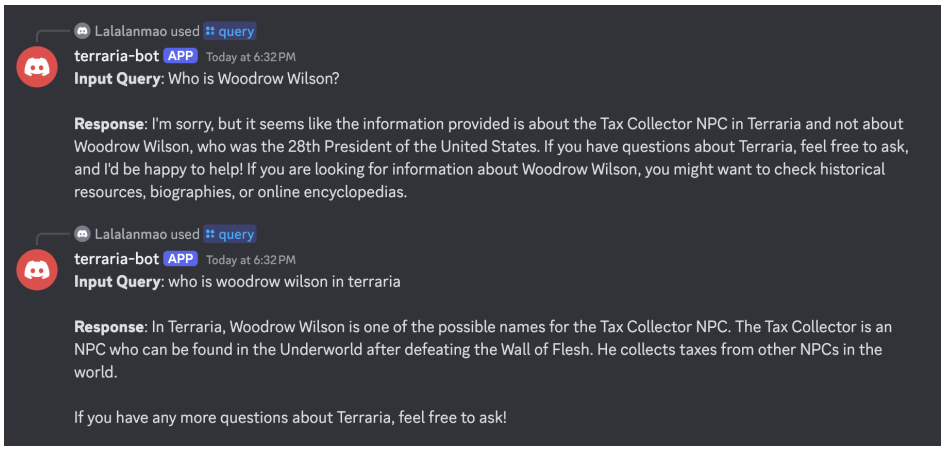
General Questions

Interesting Questions

7 Discussion/Future Improvements

Overall, I am quite happy with the results of the chatbot compared to the starting responses. However, there are some improvements that could still be made.

For example, the models I am using are all pretrained for general knowledge due to the time and cost to pretrain the models for Terraria. However, this has the draw back of queries especially regarding terms that have a more popular meaning outside of Terraria such as "Woodrow Wilson", the generation tended to generate the response in regards to the President of the US rather than toward its meaning in Terraria until the user prompt specifically included in Terraria. Some possible ways to address this is in the most efficient way is to continue working on the prompt to ensure that even without user input or identify common cases where the game information is overshadowed by its pretrained information.



Another area for improvement would be in the responses due to the extra words in the prompt, the generation would sometimes also respond to those such as with "Remember that the information provided is based on the context you've shared, and the game mechanics may slightly differ in various versions of Terraria. If you need more detailed information or encounter any issues, consider referring to the Terraria Wiki or community forums for additional guidance."

8 Past Iterations

1. **Initial Proof of Concept:** The first iteration served as a proof of concept, where a simple prototype was developed to test the model's feasibility. The process involved converting HTML directly to Markdown, and then converting each Markdown section to JSON. The embedding model used was `all-MiniLM-L6-v2`, and FAISS was employed to create an index for efficient similarity search. For query responses, the model retrieved the top five most relevant matches and generated a response using GPT-2. This approach demonstrated that while the model could handle paragraph information fairly well, it struggled significantly with table data. Additionally, due to the prevalence of bulleted lists with short phrases on the Terraria Wiki, the model's responses often included similarly structured, short responses.

2. **List and Table Processing:** To address the table-related issues, a second iteration focused on better handling of lists and tables. A script was created to convert list items into single comma-separated strings and reformat Markdown tables into plain-text structures, where each row followed the format "`col1: row1.1, col2: row1.2, ...`". This iteration revealed two key insights: (1) ChatGPT-2's token limit of 1200 tokens was restrictive. This limit had to account for

the prompt, the retrieved context, and the generated output. As a result, the retrieved text often had to be truncated, leading to incomplete information being fed into the model.

3. **Targeted Section Extraction:** In the next attempt, the focus shifted to selecting and extracting specific page sections that were most relevant, such as infoboxes (to process non-standard tables), general text sections, notes, tips, history, and specific tables like recipes and drop information. This targeted approach improved context relevance, but it revealed a new issue. While queries about general topics (like bees) produced reasonable results, queries containing more specific, niche terms (like "Cthulhu") led to repetitive outputs. Additionally, similar to issues encountered in earlier coursework with GPT-2, the model's responses frequently repeated sentences, often looping on a single idea after a few good initial sentences. Furthermore, I also noticed when looking at what was retrieved, that the history information tended to get retrieved which also affected the ability to generate accurate/relevant information. This informed my decision to remove version information as during a normal interaction, the player does not care too much about past information.

4. **Switch to GPT-3 Turbo:** Given the limitations of GPT-2—including token constraints, repetitive responses, and lengthy training times—a shift to GPT-3 Turbo was made. This decision was also informed by comparing the token density of Terraria's content to that of the much smaller training datasets (like Tiny Stories) that GPT-2 was designed to handle. The switch to GPT-3 Turbo significantly improved the quality of generated responses while reducing the need for extensive training and tuning.

5. **Comprehensive Preprocessing and Final Refinement:** In the final iteration, the preprocessing strategy was overhauled. Instead of processing distinct page categories separately (an approach found to be inefficient), a universal preprocessing method was applied to all pages. Priority was given to extracting key sections with detailed, structured preprocessing rules. Sections with minimal or list-based content were either processed using a general script or excluded entirely if the information was not essential. This change, combined with GPT-3 Turbo's improved capabilities, yielded significantly better generation results. The final steps involved refining the prompt used for GPT-3 Turbo and optimizing how the retrieval system weighted and prioritized relevant information. Further details on these refinements are discussed in the main report.

6. *I realized after I had already deleted the old preprocessing jsons and scripts to save space that I should have kept them or their outputs to show the improvement of the results in the report; however I did not and I hope this helps give an idea of the development of the model to the state in the report.*