NLP Project Report

Project:
A simple text-based game that incorporates NLP concepts into the game. These concepts allow for detailed player interactions with the game while properly displaying how their actions affect the game.

GitHub Repository: https://github.com/cas190006/NLPGameProject.git

Lessons Learned:
Learned the basics of NLP while managing a small team to produce the project. While the final project may not be as complex or extravagant as other projects, it was a passion project that we were interested in pursuing and experimenting with.

Contributions:

- Cesar Saenz

I created the base game and the player input parser. The base game is a simple dungeon explorer, in which the player must navigate it and its obstacles in order to escape. The dungeon is full of twists and turns as well as monsters that the player must fight to advance and chests full of items or weapons for the player to store or use. The input parser is what understands a player's input by identifying actions and direct/indirect objects. Given a dictionary with meanings for the words, it looks at the actions and objects to determine what the intended action is with its corresponding objects to perform the player's desired result and properly advance the game. There's also a typo checker that fixes any typo mistakes that the player might make. This is a fairly simple but effective NLP method for what we're going for.

In terms of player input, the player has a wide range of actions that they can perform. Players navigate the dungeon by typing where they want to go. If they're in a room where there is only one other exit, apart from the one they entered, they can choose to input its direction or simply say to

head to the next room. They have access to a map that shows the dungeon with the rooms they've visited and its contents, such as a monster or non-empty chest, if there are any. If the player enters a room with a monster, it initiates a fight with it. In fights, the player can use his weapon to attack enemies or his abilities, such as a stun attack and fireball attack, that have chances of occuring and need a cooldown before using again. If the enemy overwhelms them, they can choose to run away and escape the fight. If the player enters a room with a chest, they will find items that they can use or weapons that can replace their current weapon. They have the option of taking any items or leaving them in the chest. Some items that they can find are health potions that restore the player's health up to their max health and strength potions that make the player's weapon attacks stronger. They can use these potions at any time if they have one in their inventory. The only fight that is different, in terms of variety, is the boss battle at the dungeon's exit. The boss has abilities, unlike regular enemies, that have different effects on the player, such as frostbite damage done over time or stealing the player's health to heal himself, and can't be stunned or run away from.

- Sean O'Keefe

I created the ways to get the game to respond appropriately to player inputs. To do this, I ended up creating my own language model. I did not use an existing LLM because I did not want to have to pay for an API key and I wanted the messages to stay focused on the game. I thought that an LLM would be unnecessarily broad for this application and might get off topic. The language model I ended up creating is a simple ngram model. The data that is used for text generation is a .txt file with multiple response sentences representing each type of scenario in the game. Examples of these scenarios include encountering an enemy, using an item, or checking a map of the dungeon. Each of the sentences is preceded by a phrase in all capital letters that either represents the corresponding scenario for that sentence and/or shows how that sentence will be used in the game. This phrase can be thought of as the "prompt" and is the first token used in the ngram model (however, it is not printed in the game's output). This guides the output, and successive tokens are generated until a specified count of a

specific substring is in the output. Most of the time, these substrings are simple punctuation marks such as “.” or “?”. The count and the substring differs depending on the game scenario. The value of n used for the ngram also differs based on the scenario. The range of values n takes is from 3 to 6. I used higher n values such as 5 or 6 in many cases because any lower and the output wouldn’t always make sense, and I want the player to know what is happening. For example, if I let n = 3 for the outputs presented when the player attacks an enemy, the outputs might start mentioning the dungeon map or an irrelevant treasure chest. This language model I created may be simplistic, but it produces relevant yet varied output for the player and can easily be improved by adding to the .txt file.

I made a few other contributions besides the output generator. I added the ability for the player to select between “normal” and “hard” difficulty, made the enemies more interesting by naming them and giving them more varied weapons, and made some adjustments to the difficulty of the enemies by changing some of their stats after testing the game multiple times.

Self-Scoring (Cesar Saenz):

- 50 points - significant exploration beyond baseline: While the project and game is simple, it exceeds more than what we set out to build
- 30 points - Innovation or Creativity: We designed a simple text-based game with NLP elements, such as an input parser and text generation. This stands out compared to more conventional projects
- 10 points - discussion of lessons learned and potential improvements: While simple NLP concepts were implemented, they worked fairly well for the game we wanted to make
- 5 points - exceptional visualization/diagrams/repo: The game has a map feature that showcases where the player is within the dungeon along with generated text messages
- 5 points - earned money with the project: While no money has been earned, it does have the potential to generate revenue as a simple game with some refinement. Although, it should be noted that it would be low-priced given its simplicity, leading to low profits

Self-Scoring (Sean O'Keefe):

- 50 points - significant exploration beyond baseline: I think this project became more complex than the initial idea, especially regarding the combat in the game. Initially, the player and all enemies only had one form of offense, but eventually the player and the final enemy could choose different ways to attack. This also required more variety in the response types generated by the game during combat, which I believe I managed to pull off. Additionally, we were initially not sure we would offer two different levels of difficulty in the game.
- 30 points - Innovation or Creativity: I think our project is very unique and creative compared to most other projects from students in this course. Most other groups probably created some form of chatbot using machine learning techniques and LLMs. Our game is quite different from these and, in my opinion, offers more potential for future development. This game could be developed into a large-scale adventure with an interesting plot and a variety of characters; the foundation is in place for all of this. On a smaller scale of creativity, I tried to make some of the enemies represent creatures not typically found in fantasy settings, and if the game were expanded I would continue this trend. Finally, I made hard mode different enough from normal mode that it actually has an effect on what decisions a player might make.
- 5 points - highlighted complexity: Most of the sentences in the .txt file were written by me (some were from Cesar or modified versions of sentences from Cesar), and all of the ones I wrote were created without using an AI. For some categories of sentence with many examples, such as attacking an enemy, I used an online thesaurus to help me create more variety in the words contained in the sentences. It's not extremely complex, but I think it is enough for 5 out of 10 points.
- 10 points - discussion of lessons learned and potential improvements: The main lesson I learned from this project was that even simple tasks in a project like this can be time-consuming and more complicated than initially expected. As for potential improvements, I think there are many ways to drastically improve the game, and some

of these would not be too difficult to implement. One of these relatively easy improvements would be randomizing the game's dungeon so that enemies and items are in different locations each time the game is played. It could even be a partial randomization where some elements have random locations while others are always in the same locations. Another improvement would be simply making the game bigger. The dungeon in the game is currently small, but expanding it would allow for more variety in the scenarios encountered.

- 5 points - exceptional visualization/diagrams/repo: The game itself does not have many visuals, but I think our repo for this project is good because of its simplicity. There are only four files in the repo and no subfolders. This makes the repo easy to understand. People viewing the repo do not have to waste time and energy figuring it out.