Conversational AI Agents in Video Games using Natural Language Processing

Samith S. Shetty
Department of Computer Science
University of North Carolina at Charlotte
Charlotte, NC 28223
sshett15@charlotte.edu

Christopher T. Pitre
Department of Computer Science
University of North Carolina at Charlotte
Charlotte, NC 28223
cpitre1@charlotte.edu

Abstract

Natural Language Processing techniques have the potential to allow for dynamic NPCs. GPT-2's light weight architecture can help perform this task while allowing a large range of hardware to run the model. A possible way to generate dynamic dialogue is through training the model with question and answer pairs, where the model will then generate answers. Additionally, using classification similar to sentiment analysis, NLP models can be used to analyze user input without the need for complex rules to analyze text with. Generation with GPT-2 seems to require more experimentation, but classification using GPT-2 seems to be a viable format for dynamic NPCs.

1 Problem Justification

In modern video games, there have been increasing demands for engaging and novel gameplay mechanics that can help elevate the medium. Historically, expert systems have long been the dominant approach to implementing NPCs and AI agents inside of games, but this approach has a notable limitation in regards to player-to-NPC dialogue. Expert systems can only provide limited dialogue options for players to select, decreasing the amount of immersion and freedom that players have in such games. However, modern Natural Language Processing techniques have the potential to enable completely free player dialogue and increase the storytelling capabilities of games through dynamic-conversations, allowing NPCs to feel less scripted and in-game interactions to feel more realistic.

2 Methodology Development & Justification

2.1 Model

The NLP model of choice for this problem is OpenAI's GPT-2 model; specifically, Hugging Face's implementation of GPT-2 on PyTorch. The choice to build on top of GPT-2 rather than notably more capable large language models came out of performance constraints, as GPT-2 is small enough to be locally run inside of the game and generate and generate responses within reasonable time (< 4 seconds) on accessible hardware (without a GPU). The ultimate goal is for less capable hardware to be able to run a local version of the model in order to remove any need for network connectivity when playing a game while also decreasing the amount of time needed for each response. In order to optimize GPT-2 for the purposes of this problem, transfer learning and fine tuning will be used to allow for the agent to respond properly to user input.

2.2 Game Engine

The game itself was built using the Godot Game Engine. When the game is first loaded in, it spawns a separate python process which runs the trained model. This process communicates bi-directionally with the game process through UDP, receiving a prompt when the player sends a message to an NPC, and sending back the model's generated response to be displayed as the NPC's dialogue.

2.3 Method 1: Question and Answer Pair Generation

The first method chosen for creating dynamic-conversations between players and NPCs was to create question and answer pairs where a player would ask the NPC a question, and the NPC would complete a response based on the player's question. An example of what possible outputs would be is shown below:

Table 1: Question and Answer Pairs

Question	Answer
Where is the key?	The key is behind a tree.
Where am I?	You're in the Kingdom of Eldoria.
Who are you?	I am an oracle statue.

In order to fine-tune the GPT-2 model, responses such as the ones given above were staged in a PyTorch dataset where they are tokenized to have the following format:

<bos>Question: Where is the key? Answer: The key is behind a tree.<eos>

Figure 1: Example tokenized question and answer pair

This dataset contains each of the line's input IDs and attention masks. The overall split between training and validation data is 90% training data, and 10% validation data. Using a customized PyTorch training script, the GPT-2 model is then retrained using this dataset. The optimizer used is the AdamW optimizer. AdamW was chosen due to its advantages over traditional implementations of Adam, and due to the complexity of the GPT-2 Architecture [1]. In addition to AdamW, a linear scheduler with warmup was used to modify the learning rate during training. This was used to help provide more stable training, faster convergence, and better generalization [2]. The hyperparameters chosen for the training loop are as follows.

Table 2: Hyperparameters

Hyperparameter	Value
Epochs	5
Learning Rate	0.0005
Warmup Steps	100
Epsilon	0.00000001

Afterwards, using the fine-tuned GPT-2 model, the player will be able to ask questions and get answers in context of the game world.

2.4 Method 2: Classification based on Sentiment Analysis

Sentiment analysis is a sub-field in natural language processing which reads people's opinions towards a certain entity [3]. Essentially, an NLP agent is able to figure out a person's disposition towards a topic, for example, a negative or positive disposition, through identifying and classifying sentiments. Taking this idea further, using transfer learning to fine-tune GPT-2, GPT-2 can be used for sentiment analysis. Instead of classifying positive, negative, or neutral sentiments, however, GPT-2 will be used to identify and label sentences based on certain classes which will then be used to give responses to the player.

A majority of the decisions made in the previous method still stand for this method, however, the dataset being used is slightly different alongside the end goal of this method. The dataset is full of single question sentences, which are also classified by a certain class:

"Where is the key?", 0

Figure 2: Question and Class Pair

The classes that will be used for this method is as follows:

Table 3: Classes to Identify with GPT-2

Class Number	Class Meaning
0	key
1	door
2	location
3	identity
4	small talk
5	goal

This list is by no means exhaustive, and theoretically can be expanded to fit multiple different scenarios. Using these question and class pairs, GPT-2 will be prompted with each question and class pair to train, and will eventually be able to classify novel sentences based on what it has learned about each class.

This method provides advantages over the previous method by being more predictable and stable. In addition to being more predictable, this method is more measurable in terms of performance. While the previous method falls more in line with the dynamic rationale behind NLP based agents in video games, this method is a more feasible implementation of the idea which still has the potential for expansion.

3 Experiments & Results

3.1 Method 1

Method 1 yielded somewhat disappointing results upon first implementation. For simple question and answer pairs, depending on what the question was and how the model was trained, it would either generate in loops, or overfit towards certain keywords.

Table 3: Example Generated Question and Answer Pairs

Question	Answer
Where is the key?	住来是
Where am I?	My kingdom is near secure; and the key is hidden beneath a pile of leaves.
How do I leave?	I am the journey, my journey.
Who are you?	The supreme cosmic consciousness holds sway over the cosmos.:?
Where is the key?	The key is in the corner of a corner.
Who are you?	My name is Amoria.

3.2 Method 2

Method 2 was able to generate and identify classes from user inputted sentences.

Table 4: Example Generated Question and Class Pairs

Question	Class
Where am I?	location
Where is the key?	key
How is the weather?	location
Who are you?	identity

After training, these were the metrics for Method 2:

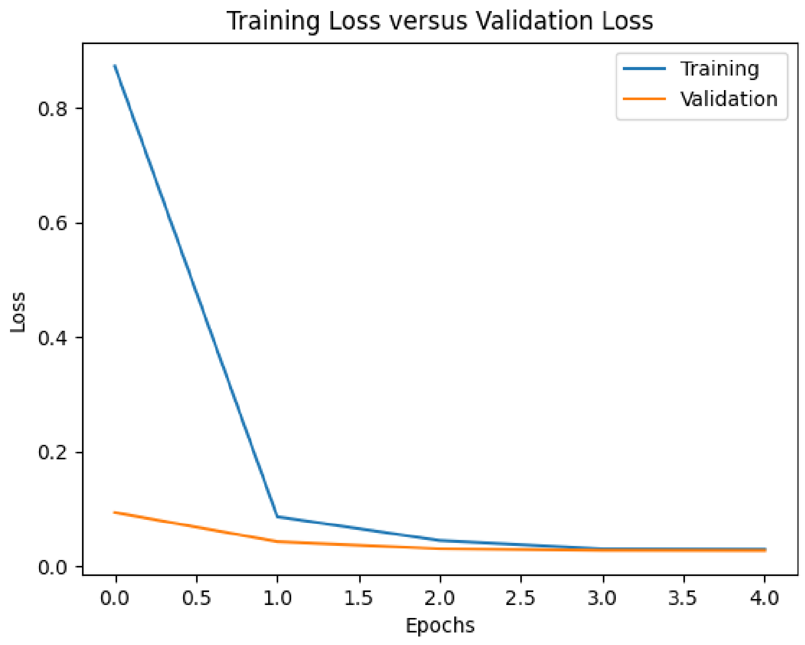

Figure 3: Training Loss versus Validation Loss

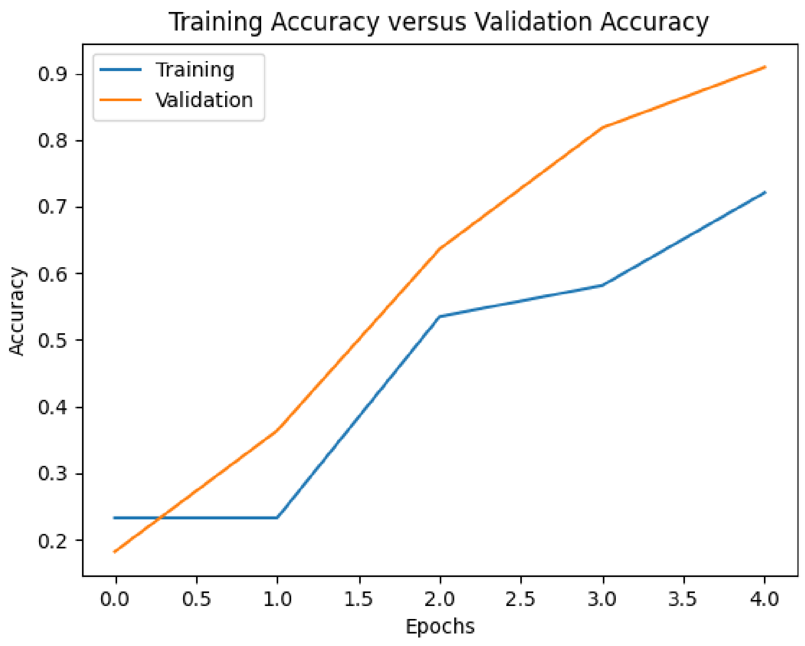

Figure 4: Training Accuracy versus Validation Accuracy

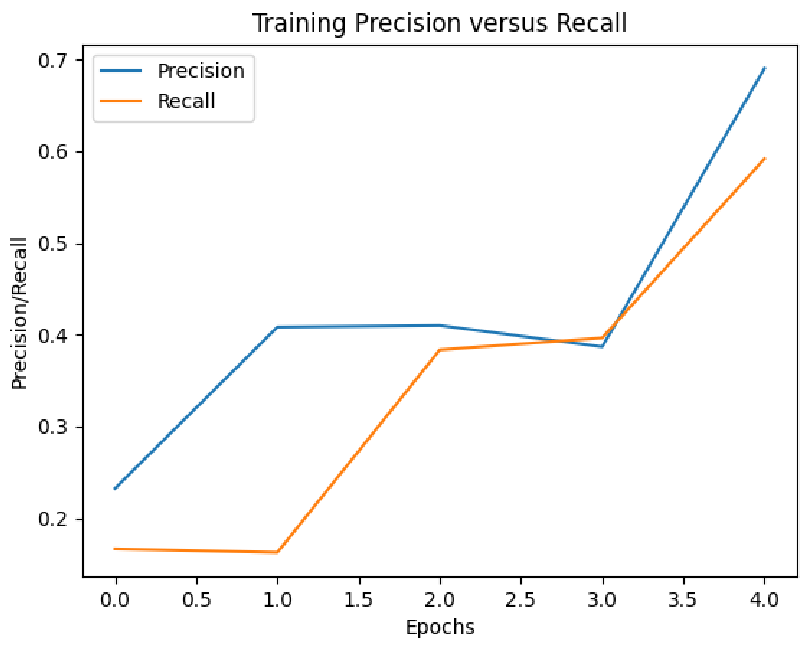

Figure 5: Training Precision and Recall

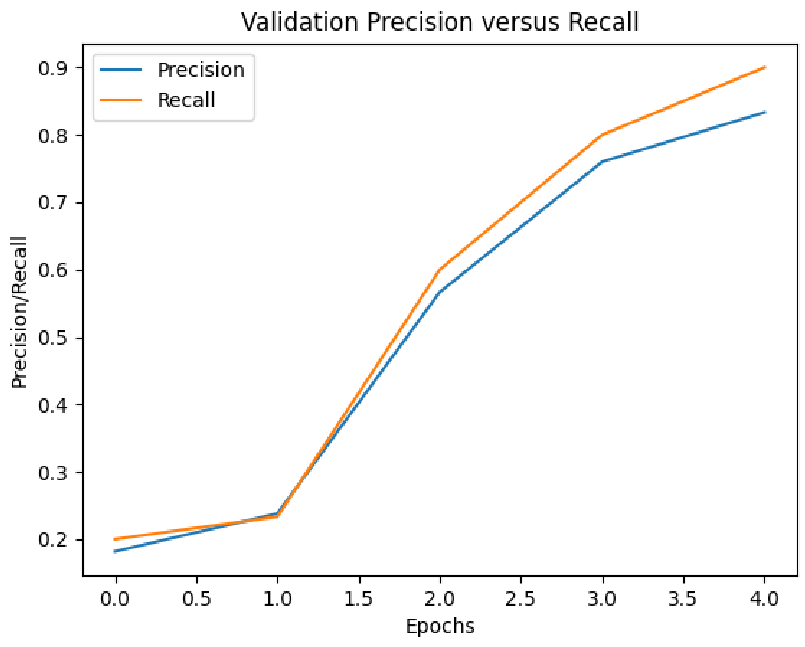

Figure 6: Validation Precision and Recall

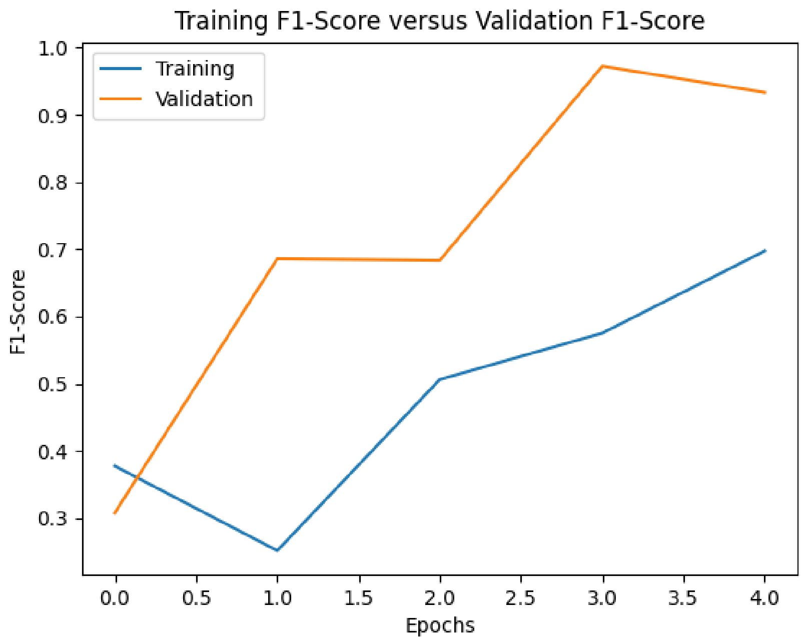

Figure 6: Training F1-Score versus Validation F1-Score

4 Discussion

The generative approach showed mixed results. Prior to fine-tuning, GPT-2 had very poor responses and would consistently get stuck in strange cycles, repeating the same phrase over and over again until hitting the max response length. After fine tuning with specific examples, we were able to eventually generate responses that were relevant to the question, but we still observed considerable noise in responses. Often they would miss the question or occasionally produce a response that devolved into incoherence part-way through. There was also a delicate balance between providing it training data that was relevant to the information it needed to tell the player, but also teaching it to say other things (for example, the model would respond with information about a hidden key without any mention of it in the prompt)

We tested using larger versions of GPT-2, but found that the computation cost, training time, and response time increased considerably while the quality of responses showed limited improvements. Having seen these results, we opted to test out a sentiment analysis approach which matched player responses to the type of information they wanted to receive. While this approach ultimately limits the versatility of the chat agent and its responses, it more simply creates high quality responses and while providing the novelty of removing explicit dialogue options for the player. We consider this to be a middle-step between the current status-quo of expert system-based AI and the future goal of fully generative models powering NPC responses. Despite this, the sentiment analysis approach still seems to be able to grow dynamically and be applied further.

Ultimately, one of the biggest limiting factors for this experiment was the size of the dataset. Given more data, the model's effectiveness would have increased. An experiment to use an exterior dataset was attempted, but the training times even with Google Colab's T4 GPUs was too long to process. For future steps, more exploration into other LLMs will should be made, and also more consideration as to fine-tuning responses with context of the surrounding environment should be made.

References

[1] Z. Zhuang, M. Liu, A. Cutkosky, and F. Orabona, “Understanding AdamW through Proximal Methods and Scale-Freeness,” arXiv (Cornell University), Jan. 2022, doi: https://doi.org/10.48550/arxiv.2202.00089.

[2] L. Liu et al., “On the Variance of the Adaptive Learning Rate and Beyond,” arxiv.org, Aug. 2019, doi: https://doi.org/10.48550/arXiv.1908.03265.

[3] W. Medhat, A. Hassan, and H. Korashy, “Sentiment analysis algorithms and applications: A survey,” Ain Shams Engineering Journal, vol. 5, no. 4, pp. 1093–1113, Dec. 2014, doi: https://doi.org/10.1016/j.asej.2014.04.011.