modifed-dm

November 17, 2024

0.0.1 Importing Libraries and Loading Model

[88]:
```
import random
import torch
from transformers import GPT2LMHeadModel, GPT2Tokenizer

# Load the fine-tuned tokenizer and model
model_name = 'gpt2-large'
tokenizer = GPT2Tokenizer.from_pretrained(model_name)
model = GPT2LMHeadModel.from_pretrained(model_name)

# Set padding token to EOS
tokenizer.pad_token = tokenizer.eos_token
```

0.0.2 Character Stats Generation

[89]:
```
# Function to roll 4d6 and drop the lowest
def roll_stat():
    rolls = [random.randint(1, 6) for _ in range(4)]
    rolls.remove(min(rolls))
    return sum(rolls)

# Function to generate stats with race and class bonuses
def generate_stats(race, char_class):
    stats = {
        "strength": roll_stat(),
        "dexterity": roll_stat(),
        "constitution": roll_stat(),
        "intelligence": roll_stat(),
        "wisdom": roll_stat(),
        "charisma": roll_stat(),
    }

    # Hardcoded race modifiers
    race_modifiers = {
        "Elf": {"dexterity": 2, "wisdom": 1},
        "Dwarf": {"constitution": 2, "wisdom": 1},
```

```
        "Human": {"strength": 1, "dexterity": 1, "constitution": 1,␣
↪"intelligence": 1, "wisdom": 1, "charisma": 1},
        "Halfling": {"dexterity": 2, "charisma": 1},
        "Dragonborn": {"strength": 2, "charisma": 1},
        "Tiefling": {"intelligence": 1, "charisma": 2},
        "Gnome": {"intelligence": 2, "dexterity": 1},
        "Half-Elf": {"dexterity": 1, "charisma": 2},
        "Half-Orc": {"strength": 2, "constitution": 1},
        "Tabaxi": {"dexterity": 2, "charisma": 1},
        "Aasimar": {"charisma": 2, "wisdom": 1},
        "Goliath": {"strength": 2, "constitution": 1},
    }

    # Hardcoded class stat preferences
    class_stat_priorities = {
        "Wizard": {"intelligence": 2},
        "Fighter": {"strength": 2, "constitution": 1},
        "Rogue": {"dexterity": 2, "charisma": 1},
        "Cleric": {"wisdom": 2, "charisma": 1},
        "Paladin": {"strength": 2, "charisma": 1},
        "Ranger": {"dexterity": 2, "wisdom": 1},
        "Bard": {"charisma": 2, "dexterity": 1},
        "Warlock": {"charisma": 2, "intelligence": 1},
        "Monk": {"dexterity": 2, "wisdom": 1},
        "Barbarian": {"strength": 2, "constitution": 1},
        "Druid": {"wisdom": 2, "intelligence": 1},
    }

    # Apply race modifiers
    if race in race_modifiers:
        for stat, bonus in race_modifiers[race].items():
            stats[stat] += bonus

    # Apply class stat preferences
    if char_class in class_stat_priorities:
        for stat, bonus in class_stat_priorities[char_class].items():
            stats[stat] += bonus

    return stats
```

0.0.3 Character Creation

[90]:
```
def create_character():
    # Display available options
    races = ["Elf", "Dwarf", "Human", "Halfling", "Dragonborn", "Tiefling",␣
↪"Gnome", "Half-Elf", "Half-Orc", "Tabaxi", "Aasimar", "Goliath"]
```

```
    classes = ["Wizard", "Fighter", "Rogue", "Cleric", "Paladin", "Ranger",␣
↪"Bard", "Warlock", "Monk", "Barbarian", "Druid"]
    alignments = ["Lawful Good", "Neutral Good", "Chaotic Good", "Lawful␣
↪Neutral", "True Neutral", "Chaotic Neutral", "Lawful Evil", "Neutral Evil",␣
↪"Chaotic Evil"]

    print("Available Races:", ", ".join(races))
    race = input("Choose your race: ")

    print("\nAvailable Classes:", ", ".join(classes))
    char_class = input("Choose your class: ")

    print("\nAvailable Alignments:", ", ".join(alignments))
    alignment = input("Choose your alignment: ")

    name = input("\nEnter your character's name: ")
    backstory = input("\nWrite your character's backstory: ")

    print("\nDescribe your character's appearance (height, build, eye color,␣
↪hair color, distinguishing features):")
    appearance = input("Appearance: ")

    stats = generate_stats(race, char_class)

    character = {
        "name": name,
        "race": race,
        "class": char_class,
        "alignment": alignment,
        "backstory": backstory,
        "appearance": appearance,
        "stats": stats,
    }

    print("\nCharacter Details:")
    print(f"Name: {character['name']}")
    print(f"Race: {character['race']}")
    print(f"Class: {character['class']}")
    print(f"Alignment: {character['alignment']}")
    print(f"Backstory: {character['backstory']}")
    print(f"Appearance: {character['appearance']}")
    print(f"Stats: {character['stats']}")

    return character
```

0.0.4 Extracting Details From Backstory

[91]:
```
import spacy
nlp = spacy.load("en_core_web_sm")

def extract_details_from_backstory(backstory):
    doc = nlp(backstory)
    details = {"locations": [], "objects": [], "themes": []}

    for ent in doc.ents:
        if ent.label_ in ["GPE", "LOC"]:  # Geographic locations or specific␣
↪settings
            details["locations"].append(ent.text)
        elif ent.label_ in ["NORP", "ORG", "EVENT", "OBJECT"]:  # Objects,␣
↪events, or groups
            details["objects"].append(ent.text)

    for token in doc:
        if token.pos_ == "NOUN" and token.text.lower() not in␣
↪details["objects"]:
            details["objects"].append(token.text)


    return details
```

0.0.5 Reconstructing Prompt From Extracted Details

[92]:
```
def construct_prompt_from_backstory(backstory, extracted_details):
    prompt = f"{backstory}\n"

    if extracted_details["locations"]:
        prompt += "The setting takes place in: " + ", ".
↪join(extracted_details["locations"]) + ". "
    if extracted_details["objects"]:
        prompt += "Key elements in the setting include: " + ", ".
↪join(extracted_details["objects"]) + ". "

    prompt += "\nDescribe this setting in vivid detail:"
    return prompt
```

0.0.6 Setting Generation

[93]:
```
def generate_dynamic_setting(character):
    prompt = (
        f"{character['name']} is a {character['alignment']} {character['race']}␣
↪{character['class']}. "
```

```
        f"{character['name']} Backstory: {character['backstory']}. "
        f"{character['name']} attributes include - Strength:␣
↪{character['stats']['strength']}, Dexterity:␣
↪{character['stats']['dexterity']}, "
        f"Constitution: {character['stats']['constitution']}, Intelligence:␣
↪{character['stats']['intelligence']}, "
        f"Wisdom: {character['stats']['wisdom']}, Charisma:␣
↪{character['stats']['charisma']}.\n\n"
        "Setting description:"
    )

    # Tokenize the prompt
    inputs = tokenizer(prompt, return_tensors="pt", padding=True,␣
↪truncation=True, max_length=1024)
    attention_mask = inputs["input_ids"] != tokenizer.pad_token_id

    # Generate setting with the language model
    outputs = model.generate(
        inputs["input_ids"],
        attention_mask=attention_mask,
        max_new_tokens=250,
        temperature=0.7,
        top_k=50,
        top_p=0.9,
        do_sample=True,
        repetition_penalty=1.2,
        pad_token_id=tokenizer.eos_token_id,
        eos_token_id=tokenizer.eos_token_id
    )

    # Decode and return the generated setting
    setting = tokenizer.decode(outputs[0], skip_special_tokens=True).strip()
    return setting
```

0.0.7 Initiating Adventure with Character and Setting

[94]:
```
def start_game_narration(setting, character):
    prompt = (
        f"The adventure begins in the setting:\n\n"
        f"{character['name']} is a {character['alignment']} {character['race']}␣
↪{character['class']}. "
        f"{character['name']} now finds themselves at the center of unfolding␣
↪events and prepares for what comes next\n\n"
        f"Dungeon Master: As {character['name']} steps into the unknown, the␣
↪story begins. What will {character['name']} do?"
    )
```

```
    print(prompt)
```

0.0.8 Game Setup

[95]:
```
# Create a character and generate a setting
character = create_character()
extracted_details = extract_details_from_backstory(character['backstory'])
print(extracted_details)
character['backstory'] =␣
 ↪construct_prompt_from_backstory(character['backstory'],extracted_details)

setting = generate_dynamic_setting(character)

# Split the prompt into two parts: before and after "Setting description:"
parts = setting.split("Setting description:")
character_info = parts[0].strip()
setting_description = parts[1].strip() if len(parts) > 1 else ""


# Dungeon Master's introduction and start of the game
print("\nDungeon Master: Here is your setting\n")
print(setting_description)
print("\nThe game begins...\n")
start_game_narration(setting, character)
```

```
Available Races: Elf, Dwarf, Human, Halfling, Dragonborn, Tiefling, Gnome, Half-
Elf, Half-Orc, Tabaxi, Aasimar, Goliath

Available Classes: Wizard, Fighter, Rogue, Cleric, Paladin, Ranger, Bard,
Warlock, Monk, Barbarian, Druid

Available Alignments: Lawful Good, Neutral Good, Chaotic Good, Lawful Neutral,
True Neutral, Chaotic Neutral, Lawful Evil, Neutral Evil, Chaotic Evil

Describe your character's appearance (height, build, eye color, hair color,
distinguishing features):

Character Details:
Name: Scorio
Race: Human
Class: Fighter
Alignment: Lawful Good
Backstory: Scorio was once known as the lord of Nagaran. He was the king of
Runeterra. He was a just king and the people loved him
Appearance: tall, dark and handsome
Stats: {'strength': 11, 'dexterity': 12, 'constitution': 15, 'intelligence': 14,
'wisdom': 17, 'charisma': 16}
```

{'locations': ['Nagaran'], 'objects': ['king', 'people'], 'themes': []}

Dungeon Master: Here is your setting

The world of Runeterra is in chaos after the death of its king. It has been ruled by two rulers since then; one from Runeterra's east and one from Runeterra's west. One person believes that they are responsible for the turmoil, but the other believes that it is the work of the demon Noxus. Both believe that the king's daughter, Shurima, should rule. The ruler of Nagarand has also decided to try and take over Runeterra. There have been many wars between the two rulers. A small band of heroes who have come together to help each other fight the evils plaguing their kingdom and save their people from being lost forever. They are the Heroes of Justice. (Sic)

The Setting is a fantasy setting with some fantasy elements. This setting is a mixture of fantasy settings and sci-fi settings.

The game begins…

The adventure begins in the setting:

Scorio is a Lawful Good Human Fighter. Scorio now finds themselves at the center of unfolding events and prepares for what comes next

Dungeon Master: As Scorio steps into the unknown, the story begins. What will Scorio do?