

Fantasy AI





Funkcje

1. Automatyczne streszczanie świata fantasy z pliku.
 2. Generowanie NPC zgodnie ze schematem JSON (imię, frakcja, profesja, cechy, krótka historia).
 3. Pobieranie wygenerowanego NPC jako pliku JSON.
 4. Rozmowa gracza z wybranym NPC w stylu czatu.
 5. Symulacja rozmowy NPC vs NPC (naprzemienne dialogi).
-

Technologie

1. Streamlit – interfejs webowy.
 2. LM Studio API – lokalne LLM.
 3. Requests – komunikacja z modelem.
 4. JSON – zapis NPC.
 5. Python i jego standardowe biblioteki.
-

Instalacja

1. Pobierz projekt.
 2. Zainstaluj LM Studio (lmstudio.ai/)
 3. Pobierz i uruchom wybrany model (polecany i przetestowany to mistral-7b-instruct-v0.3) – pobrania można dokonać samemu z huggingface.co lub łatwo za pomocą lmstudio (zalecane).
 4. W pliku game_system.py upewnij się, że MODEL_NAME odpowiada uruchomionemu modelowi (np. "mistralai/mistral-7b-instruct-v0.3").
 5. Nawiguj CMD do katalogu projektu.
 6. Zainstaluj wymagane pakiety komendą: `pip install -r requirements.txt`
 7. Uruchom aplikację komendą: `python -m streamlit run system.py`
 8. W przeglądarce pod adresem <http://localhost:8501> otworzy się panel z zakładkami:
 -  Story Q&A – pytania o świat.
 -  NPC Generator – tworzenie nowych NPC.
 -  Talk with NPC – rozmowa z wybranym NPC.
 -  AI vs AI – symulacja rozmowy NPC vs NPC.
-

Struktura

- game_system.py – główny skrypt aplikacji.
 - fantasy.md – plik z lore świata (do streszczania).
 - NPCs/ – katalog na zapisanych NPC w formacie JSON.
 - styles.py – style HTML/CSS czatu.
-

Konfiguracja

- Adres API LM Studio ustawiony na <http://127.0.0.1:1234/v1>.
 - Zmiana modelu możliwa przez zmienną środowiskową LMSTUDIO_MODEL lub edycję MODEL_NAME.
 - Parametry generowania (temperature, max_tokens) można regulować w kodzie.
-

Uwagi

- Generator NPC wymusza poprawny JSON – w razie błędu trzeba ponowić generację.
 - Historia czatu NPC zapisywana jest w st.session_state – resetuje się po odświeżeniu strony.
-

Eksport

- Każdego NPC można pobrać jako plik JSON i zapisać w katalogu NPCs/, aby później używać go w rozmowie.
-

Q&A

- Dlaczego wykorzystuje model lokalny?
 - Jest to darmowe rozwiązanie. Jest też bardziej elastyczne w doborze modeli AI do konkretnych zastosowań – np. model po fine-tuningu wyspecjalizowany w roleplay w świecie fantasy.
- Dlaczego wybrałem LM Studio?
 - Pomijając zaawansowanie a zarazem prostotę tego programu – posiadam obecnie (stety/niestety) grafikę marki AMD. Nie mam więc możliwości korzystania z technologii CUDA. W celu przyspieszenia działania modeli (w stosunku do działania na CPU) oraz zwiększenia kompatybilności wykorzystuje programy obsługujące technologię Vulkan – która pozwala na wykorzystanie GPU wszystkich producentów (Nvidia, AMD, Intel) do obsługi modeli AI. LM Studio posiada taką obsługę.
- Skąd pomysł na „Talk with NPC” oraz „AI vs AI”?
 - Zadanie bardzo mnie zainteresowało. Po ukończeniu „podstawowych” wymagań zadania, uznałem, że można włożyć stosunkowo niewiele dodatkowej pracy by umożliwić również rozmowę z wygenerowanymi NPC. Kolejnym eksperymentem było stworzenie możliwości rozmowy ze sobą dwóch NPC-ów.

- Dlaczego wykorzystuje ten konkretny model LLM?
 - Z tym modelem mistral miałem już do czynienia w przeszłości. Dotychczas okazywał się dla mnie najlepszym kompaktowym modelem LLM – zwłaszcza przy generowaniu kodów SQL czy JSON, a także roleplay i innych zastosowań.
- Jakie napotkano problemy?
 - Na etapie realizacji podstawowej części zadania nie pojawiły się większe trudności. Poważniejszy problem wystąpił dopiero przy próbie wdrożenia funkcjonalności „AI vs AI”. Początkowy pomysł zakładał uruchomienie dwóch oddzielnych modeli LLM prowadzących ze sobą dialog. Takie rozwiązanie wymagałoby znacznych zasobów obliczeniowych i pamięci RAM/GPU. Po krótkiej analizie możliwości przyjąłem alternatywne podejście – wykorzystanie jednego modelu, który na zmianę generuje wypowiedzi z perspektywy npc1 i npc2. Pojawił się jednak nowy problem: odpowiednie skonstruowanie promptów. Model miał bowiem tendencję do generowania całych fragmentów dialogu (obie strony jednocześnie), zamiast pojedynczej, oczekiwanej wypowiedzi.
- Plany na rozwój w krótkim zakresie czasu?
 - Dodanie więcej pól do generowanych postaci np. rasa, wiek, płeć – pozwoliłyby na znacznie większą immersję w przypadku rozmowy z postacią lub rozmowy postaci między sobą.
 - Poprawa szaty graficznej – obecna jest dosyć podstawowa.
 - Dodanie do rozmowy z NPC oraz AI vs AI znajomości innych wygenerowanych postaci ze świata gry (obecnie znają one jedynie historię świata z pliku `fantasy.md`).
- Plany na rozwój w dłuższym zakresie czasu?
 - Dodanie modelu typu image generation do generowania obrazów postaci
 - Dodanie statystyk (w tym punktów życia) postaci – w przypadku potyczki w trybie AI vs AI mogłyby one być wykorzystywane do obliczania zadawanych obrażeń. System obliczania obrażeń powinien być jednak możliwie mocno przeniesiony z modelu LLM na skrypt – model LLM poza opisem tekstowym przekazywałby wskazywałby np. szybkość, siłę i lokalizację trafienia. Informacje te trafiałyby do skryptu, który następnie obliczałby obrażenia obniżone np. o pancerz danej postaci. Modele LLM często mylą się w obliczeniach i bez sensu je dodatkowo obciążać.
 - Wybór lepszego modelu i jego fine-tuning.
 - Dodanie modelu walidacyjnego, który analizowałby rozmowy z LLM i potrafił (w połączeniu z kodem):
 - Zatrzymać i ponowić generację wiadomości przez główny LLM np. gdy wychodzi ze swojej roli.
 - Zachęcić główny model LLM do ukierunkowania rozmowy w konkretnym kierunku np. gdy model walidacyjny uzna, że rozmowa jest nudna lub powtarzająca się.
 - Analizować rozmowy użytkowników z głównym modelem LLM w celu generowania zbiorów do fine tuningu modeli poprzez zestawienie wiadomości głównego modelu LLM i odpowiedzi/reakcji użytkownika (pozytywnej lub negatywnej – według oceny walidacyjnego LLM). W ten sposób publiczne udostępnienie postaci do rozmowy demo (tak jak obecnie na stronie `foregamer`) mogłoby generować znaczące ilości danych do dalszego rozwoju modeli.