PROJET -
APPRENTISSAGE PROFOND
8INF892

Paul-Emmanuel Fulgence ATTE ATTP06030200

Basil COURBAYRE COUB30119800

Ali GHAMMAZ GHAA27070100

Diane LANTRAN LAND28580000

Avril 2024

Résumé du projet

Dans ce projet nous avons tenté d'implémenter des agents dans un sandbox qui ont une logique (action, dialogue) qui est gérée par un modèle de langage. Ce projet s'inspire de l'article de recherche:

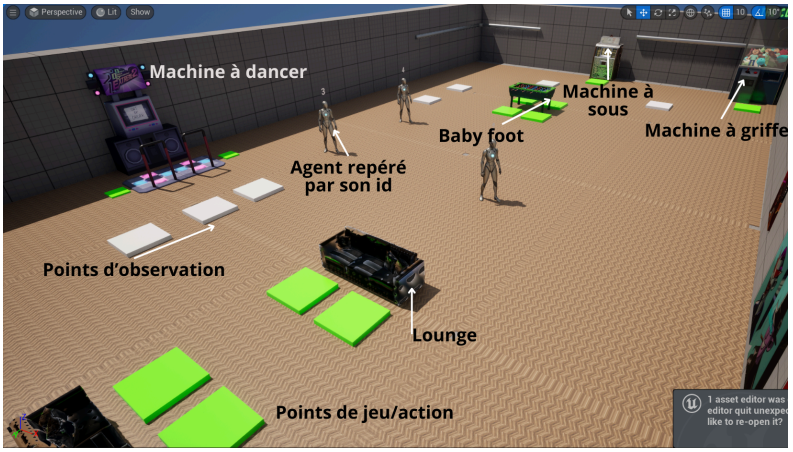
Park, Joon Sung, et al. "Generative agents: Interactive simulacra of human behavior." *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 2023.

Lors de ce projet ces différentes tâches ont été réalisées:

1. Entrainement d'un LLM de type LLama a l'aide de PyTorch et HuggingFace
2. API Endpoint à l'aide du stack LangChain (LangChain, LangServe et LangSmith)
3. Mécanisme de Mémoire des différents agents
4. Environnement de jeu dans Unreal Engine 5

Tâches réalisées

1. Environnement de Jeu Dans Unreal Engine 5



Le Jeu est accessible via le dépôt PERFORCE
L'exécutable a également été publié sur la page
https://dianelantran.itch.io/deeplsandbox

Développement d'un environnement de salle d'arcade dans laquelle des NPCs peuvent interagir entre-eux faire connaissance, observer les autres, effectuer des actions et jouer à des jeux d'arcade. Encodage du comportement des NPCs pour décider de quelle action faire ensuite et d'un système de question-réponse utilisant un LLM de type llama2.

2. API Endpoint l'aide du stack LangChain

- Afin de pouvoir intégrer GPT-3 dans Unreal Engine 5 on tente de monter un serveur qui interface avec le modèle de langage GPT-3
- Pour realiser cette tache nous utilisons LangChain, LangServe et LangSmith
- L'idée est d'utiliser L'API REST pour communiquer entre le serveur et UnReal Engine 5 permettant aux agents d'avoir des actions et du dialogue généré par GPT-3

my-langserve > packages > my-app > my_app > chain.py > ...

```python
from langchain_core.prompts import ChatPromptTemplate
from langchain_openai import ChatOpenAI

_prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            "You are a helpful assistant who speaks like a pirate",
        ),
        ("human", "{text}"),
    ]
)
_model = ChatOpenAI()

# if you update this, you MUST also update ../pyproject.toml
# with the new `tool.langserve.export_attr`
chain = _prompt | _model

```

- Configuration d'une application simple avec LangChain ou le modèle est demandée de répondre comme un Pirate

```
PS C:\Users\paule\OneDrive\Paul-Emmanuel\UQAC\Session 2\8INF892 - Apprentissage Profond\Project\langchain-zero-to-hero-0\my-langserve> langchain serve
INFO:     Will watch for changes in these directories: ['C:\\Users\\paule\\OneDrive\\Paul-Emmanuel\\UQAC\\Session 2\\8INF892 - Apprentissage Profond\\Project\\langchain-zero-to-hero-0\\my-langserve
ro-0\\my-langserve']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [19868] using StatReload
INFO:     Started server process [7532]
INFO:     Waiting for application startup.

 __          ___      .__   __.  _______      _______. _______ .______     ____    ____  _______
|  |        /   \     |  \ |  | /  _____|    /       ||   ____||   _  \    \   \  /   / |   ____|
|  |       /  ^  \    |   \|  ||  |  __     |   (----`|  |__   |  |_)  |    \   \/   /  |  |__
|  |      /  /_\  \   |  . `  ||  | |_ |     \   \    |   __|  |      /      \      /   |   __|
|  `----./  _____  \  |  |\   ||  |__| | .----)   |   |  |____ |  |\  \----.  \    /    |  |____
|_______/__/     \__\ |__| \__| \______| |_______/    |_______|| _| `._____|   \__/     |_______|

LANGSERVE: Playground for chain "/my-app/" is live at:
LANGSERVE:  │
LANGSERVE:  └─> /my-app/playground/
LANGSERVE:
LANGSERVE: See all available routes at /docs/
```

- L'application est disponible via localhost:8000

Curl

```
curl -X 'POST' \
  'http://localhost:8000/my-app/invoke' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "input": {
    "text": "hi, there"
  },
  "config": {},
  "kwargs": {}
}'
```

Request URL

```
http://localhost:8000/my-app/invoke
```

Server response

- On voie la commande POST qui permet d'envoyer une requête vers le serveur, ici le message "hi, there" est envoyé

- Dans le champ “content” nous voyons la réponse du modèle

3. Entrainement d’un LLM de type LLama a l’aide de PyTorch et HuggingFace
4. Création d’une mémoire pour les agents avec un stream de mémoire qui peut stocker des mémoires en c++, mais plusieurs limitations à ce niveau dû à la difficulté d’apprentissage de Unreal, le pont entre les fichiers c++ et Unreal n’a pas pu être réalisé. Cependant, une partie des fichiers sont disponibles pour consultation.

Limitations

1. Intégration des modèles de langage dans Unreal Engine 5
2. Prise en main de Unreal Engine 5
3. Limitations matérielles des ordinateurs utilisés

Extensions du projet

1. Intégration de l’API Endpoint LangChain dans UE5
2. Généraliser l’API EndPoint avec n’importe quel modèle de langage