Le Mans School Of AI

Le sujet

Cette année, nous allons vous faire découvrir le monde merveilleux des LLM (Large Language Model).

Les LLM sont la révolution de 2022-2023 surtout connue pour ChatGPT mais c'est aussi un nouvel outil pour les développeurs informatiques. Un outil qui permet d'utiliser un LLM pour exécuter une tâche algorithmique qui serait difficile à implémenter de manière traditionnelle.

Pour l'expérimenter et apprendre à le dompter nous vous proposons d'implémenter un jeu de société très "humain" : Loup-Garou.
Loup-Garou est un jeu de société qui ne nécessite aucun matériel et se joue uniquement en discutant avec les autres participants et le maître de jeu. C'est le terrain de jeu idéal pour une intelligence artificielle (IA) à base de LLM.

Référence des règles du jeu : https://www.regledujeu.fr/loup-garou-regle/

Objectif général

Votre objectif est de produire un jeu de société jouable par un être humain contre un ou plusieurs adversaires IA qui joueront les différents rôles.

Les adversaires devront être joués par un grand modèle de langage (Large Language Model, LLM), en minimisant autant que possible l'utilisation de code traditionnel pour l'implémentation des joueurs IA.

Important !

Notre objectif est aussi de vous apprendre à maîtriser ces outils, vous devrez donc obligatoirement utiliser un modèle de LLM en opensource et l'exécuter en local ou dans un environnement de type Google Collab.
Les modèles commerciaux (OpenAI, Anthropic, etc.) sont généralement plus performants et accessibles par API payante mais ils sont en dehors du périmètre pour ce sujet.

Objectifs intermédiaires

Partie 1 : Mettre en oeuvre un LLM

Mettez en œuvre un LLM "natif" en local sur votre machine ou dans un Google Collab et fournissez-nous la capacité à interagir avec ce LLM en mode texte pour démontrer l'atteinte de cette étape.

Modalité d'évaluation :

- Montrez-nous votre code, et démo de discussion avec le LLM
- L'évaluation ne porte pas sur la beauté de votre code, ni sur le délai pour arriver à cette étape (continuez sans attendre de démontrer cette étape si nous sommes indisponibles)

Partie 2 : Connectez le LLM avec du code traditionnel

A partir d'une liste de joueurs fournie en entrée, interrogez le LLM pour lui demander de voter sur celui à tuer. Le LLM devra fournir une réponse indiquant la personne à tuer et pourquoi et vous devrez nous montrer que vous arrivez à stocker ces deux réponses dans des variables et les afficher.

Modalité d'évaluation :

- Montrez-nous cette simulation d'une étape de vote avec un LLM

Partie 3 : Partie de Loup-garou minimaliste

Démontrez une partie de Loup-garou simplifiée avec :

- 5 joueurs IA de type villageois
- 2 joueurs IA de type loup-garou
- *(pas de joueur humain pour cette première étape)*

L'IHM peut être minimaliste pour cet objectif intermédiaire, la démo en mode débug peut-être suffisante. Il est surtout attendu de nous montrer quelques exemples de prompts utilisés et la réponse récupérée auprès du LLM, la manière dont vous gérez le contexte du joueur, etc.

Partie 4 : Permettre une partie humain avec/contre plusieurs joueurs IA

Maintenant vous devez démontrer une partie de Loup-garou minimaliste (8 joueurs sauf le meneur avec 2 loups-garous, 1 voyante et 5 villageois) avec un joueur humain et le reste des rôles joués par des IA.
L'humain pourra choisir de jouer un loup-garou ou un villageois. Des points bonus seront attribués si le joueur humain peut également choisir de jouer la voyante.

Partie 5 : Fournir une IHM de jeu ergonomique

La dernière étape correspond à votre démonstration finale. Plus votre produit fini sera complet et ergonomique et plus votre note sera élevée.
Notamment et par ordre de priorité :

- Qualité du comportement des AI : cohérence de réaction dans le jeu, peu d'hallucination
- Qualité de l'IHM et de son ergonomie
- Exhaustivité des rôles supportés dans le jeu (y compris les extensions si vous le souhaitez)

Evaluation finale

L'évaluation totale s'effectue sur :

- Le produit fini et sa démonstration
- Le passage des étapes intermédiaires permettant de démontrer votre progression

Pour votre démonstration vous aurez environ 5 à 10 mn dans lesquelles il faudra

- Nous présenter de manière macro l'architecture de votre solution
- Nous faire une démonstration d'une partie avec au moins 2 tours de jeu

Conseils

LLM en local ou dans Google Collab ?

Un LLM peut tourner en local sur votre CPU ou GPU en fonction de la quantité de mémoire de votre machine. Par exemple le modèle Phi-2B avec une “quantization” en 4 bits ne requiert que 4,3 Go de mémoire (carte graphique ou mémoire traditionnelle de l’ordinateur)
Sinon vous pouvez également utiliser Google Collab qui offre la capacité à exécuter des modèles beaucoup plus massifs.

En local il sera plus simple d’accéder à votre LLM pour y associer votre IHM et construire une solution homogène sur l’ensemble de votre application. Cependant la vitesse de génération de texte pourrait être “lente” en fonction de votre machine.
A l’inverse avec un Google Collab il sera plus compliqué de mettre en œuvre une IHM, ou alors il faudra exposer une API à partir de votre Google Collab pour l’appeler dans votre application (ex. : NGrok + Flask). Mais la vitesse de génération de texte sera élevée.

En termes de vitesse de développement pour ces 24h les deux solutions s’équilibrent.
A noter que Google Collab vous limite à 12h de GPU par jour, il vous faudra donc être vigilant ou basculer sur un second compte en cours de nuit. Pensez à exporter régulièrement votre notebook.

Un LLM de quelle taille doit-on utiliser ?

La taille des LLM est mesurée en milliards de poids dans le réseau, par exemple : 7B pour 7 milliards (“billions” en anglais).
Un modèle de 2B à 7B semble suffisant pour ce sujet.
Un modèle plus grand sera généralement plus cohérent mais l’évaluation tiendra compte de la taille du modèle que vous utilisez pour ne pas créer de distorsion en fonction des ressources machine à votre disposition.

Comment sélectionner le LLM à utiliser ?

Vous pouvez vous fier au LeaderBoard de Huggingface :
https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard
ou celui-ci :
https://llm-leaderboard.streamlit.app/
Les métriques intéressantes pour ce type de besoin sont :

- MMLU : qui correspond à la performance sur la compréhension du langage
- La compréhension du français

Les modèles suivants proposent de bons compromis pour ces 24h du code :

- OpenChat-7B
- Zephyr-7B
- Phi-2B

Qu’est que la “quantization” ?

Un modèle d’IA est généralement entraîné avec des poids qui sont stockés sur 32 bits (certains modèles peuvent aussi être entraînés en 16 bits mais cela est plus rare).
Une fois le modèle entraîné son utilisation pour générer du texte, des images, des prédictions, etc ne nécessite cependant pas de garder une résolution 32 bits ou même 16 bits. On parle alors de “quantization” pour réduire la résolution des poids du modèle tout en préservant le maximum de sa performance.
L’intérêt de cette pratique est de fortement diminuer le volume de mémoire nécessaire pour exécuter le modèle, ce qui est très intéressant pour une machine de bureau ou même un téléphone.
La “quantization” n’a pas de limite et peut descendre jusqu’à 2 bits. Bien évidemment la performance du modèle est alors fortement réduite.
Les résolutions de 5 ou 6 bits sont généralement un bon compromis taille/performance pour les LLM.

En local, quel framework est recommandé ?

Il existe de nombreuses manière de lancer un LLM en local sur votre Windows / Linux / Mac, les deux principales sont :

- Le framework HuggingFace
- Le framework Llama.cpp

HuggingFace est à la fois un site web qui correspond au “GitHub des modèles d’IA” et un framework d’exécution des modèles. Vous pouvez par exemple lancer le modèle Phi-2B en suivant les explications fournies sur la page du modèle :
https://huggingface.co/microsoft/phi-2
En utilisant ce framework vous exécutez le modèle dans sa version nominale, qui peut donc nécessiter un niveau de ressources machine significatif.

Llama.cpp est un framework optimisé pour exécuter des LLM en local sur plusieurs architectures (GPU, CPU, Apple Silicon, etc.). Il supporte différents modèles en natif mais permet également de charger des modèles au format GGUF. Ce format permet d’exposer des modèles qui ont fait l’objet d’une “quantization”. Un modèle “quantizé” peut être téléchargé sur Huggingface, en particulier sur l’espace de l’utilisateur “TheBloke” qui s’est fait la spécialité de mettre à disposition systématiquement tous les modèles opensource de référence.
Par exemple, cette page : https://huggingface.co/TheBloke/phi-2-GGUF permet de télécharger le modèle Phi-2B dans l’un des niveaux de “quantization” que l’on souhaite. Il suffit alors de le mettre dans un répertoire et de le charger avec Llama.cpp pour l’exécuter.

Langchain ?

Langchain (https://github.com/langchain-ai/langchain) est un framework qui permet d'encapsuler l'usage d'un LLM.
Cela peut être un accélérateur intéressant pour la mise en œuvre de ce projet des 24h du code mais son usage n'est nullement indispensable.
Si vous l'utilisez n'oubliez pas qu'il est conçu initialement pour utiliser des LLM commerciaux à base d'API alors que votre objectif ici est de l'utiliser avec un modèle s'exécutant en local (ou dans Google Collab).