Implémentation d'un ChatBot

Florian Geillon, Antoine Billod, Stanislas Bagnol

October 6, 2023

1 Introduction

Un chatbot est un logiciel qui a pour but de de dialoguer avec un utilisateur. Une avancée majeure dans le domaine a eu lieu en automne 2022 quand l'entreprise OpenAI a donné le libre accès d'utilisation à leur chatbot nommé "ChatGPT". En février 2023, Meta, la société possédant notamment Facebook, a publié son LLM [1]("large language model") nommé LLaMa à la communauté de recherche sous licence non commerciale [2]. Moins d'une semaine après sa publication, ses poids ont été divulgués au public sur 4chan via BitTorrent. Suite à cet incident, de grandes communautés ont entrepris de reproduire en imitant le model de LLaMa et de l'ouvrir en "open source". Nous allons ici vous présenter l'implémentation de quelques modèles les plus répandus disponibles majoritairement en Open Source.

2 OpenLLaMA

La première version que nous avons expérimentée est celle proposée par OpenLM Research nommée "OpenLLaMA: An Open Reproduction of LLaMA". Sur leur github [3] il est stipulé qu'il s'agit d'une "licence permissive open source d'une reproduction de LLaMA". OpenLLaMa fournit plusieurs modèles pré-entrainés (la valeur des poids des neuronnes) plus ou moins efficaces et donc plus ou moins volumineux. Nous avons voulu essayer leur variante la plus légère de 3B nécessitant moins de puissance de calcul. Pour ce faire, il faut dans un premier temps télécharger les fichiers disponibles sur leur page Hugging Face [4]. Pour pouvoir utiliser le programme, les bibliothèques "torch" et "transformers" devront être installées comme ceci :

```
pip install torch
pip install transformers
```

Nous avons modifié le main python qu'il propose pour qu'il puisse fonctionner :

```
import torch
from transformers import LlamaTokenizer, LlamaForCausalLM

model_path = "."
tokenizer = LlamaTokenizer.from_pretrained(model_path)

model = LlamaForCausalLM.from_pretrained(
    model_path, torch_dtype=torch.float32,
    offload_folder='./offload',device_map='auto',
)

prompt = 'Q: What is the largest animal?\nA:'
input_ids = tokenizer(prompt, return_tensors="pt").input_ids

generation_output = model.generate(
    input_ids=input_ids, max_new_tokens=32
)

print(tokenizer.decode(generation_output[0]))
```

Mais avec cette implémentation, la durée de réponse du modèle était très longue, d'environ une vingtaine de minutes pour n'importe quelle question. Non avons modifié le nombre de tokens de réponse de 32 à 8 pour avoir une réponse en 5 minutes (le temps de réponse est linéaire au nombre de tokens). Voici un exemple de question et de réponse avec ce modèle :

Q: What is the largest animal ? A: The largest animal is the blue whale. It can weigh up to 200 tons and is 30 meters long.

Pour le rendre plus rapide il faudrait utiliser soit une machine plus puissante soit un serveur avec de bonnes capacités. Nous avons par exemple fait tourner le modèle sur Google Collab (donc sur les serveurs de Google), nous avons, avec ce principe, des temps de réponses pratiquement instantannés puisque Google met à disposition une grande puissance de calcul et notamment sur GPU qui est particulièrement efficace pour le Machine Learning.

3 Alpaca

On peut lire sur la page wikipédia [2] de LLaMA que "Stanford University Institute for Human-Centered Artificial Intelligence (HAI) Center for Research on Foundation Models (CRFM)" a publié Alpaca, une recette d'entraînement basée sur le modèle LLaMA 7B qui utilise la méthode d'instruction "Self-Instruct" pour acquérir des capacités comparables au modèle text-davinci-003 de la série OpenAI GPT-3.5 à un coût modeste. Plusieurs projets open source poursuivent ce travail de fine-tuning de LLaMA avec l'ensemble de données Alpaca". Nous avons notamment essayé ce projet : https://github.com/antimatter15/alpaca.cpp. L'avantage est qu'il est très facile à utiliser :

```
git clone https://github.com/antimatter15/alpaca.cpp
cd alpaca.cpp
wget https://huggingface.co/Sosaka/Alpaca-native-4bit-ggml/resolve/main/ggml-alpaca-
7b-q4.bin

make chat

./chat
```

La version ci-dessus est fonctionnelle sur Mac OS. Sur Windows, il faudrait suivre les étapes suivantes :

```
git clone https://github.com/antimatter15/alpaca.cpp
cd alpaca.cpp
wget https://huggingface.co/Sosaka/Alpaca-native-4bit-ggml/resolve/main/ggml-alpaca-
7b-q4.bin

cmake .
cmake --build . --config Release

.\Release\chat.exe
```

Nous n'avons malheureusement pas réussi à le faire fonctionner sur Windows. Nous avons uniquement testé sur une seule machine, il est possible que le problème vienne de la configuration de cette dernière. La principale différence entre LLaMa et Alpaca est l'utilisation des tokens de q4 (4 bits) au lieu de float16 (16 bits), ce qui permet un temps de réponse bien plus rapide, quasiment instantané.

4 Faraday

Faraday [5] est un site qui propose une application qui facilite le téléchargement de modèle et qui fournit directement une interface semblable à celle de chatGPT. C'est un client LLM [1]. Le nombre de modèles disponibles est limité car ils font l'objet de vérification par les développeurs de l'application. Les modèles disponnibles sont par exemple Vicuna, Wizard ou encore Vigogne en français.

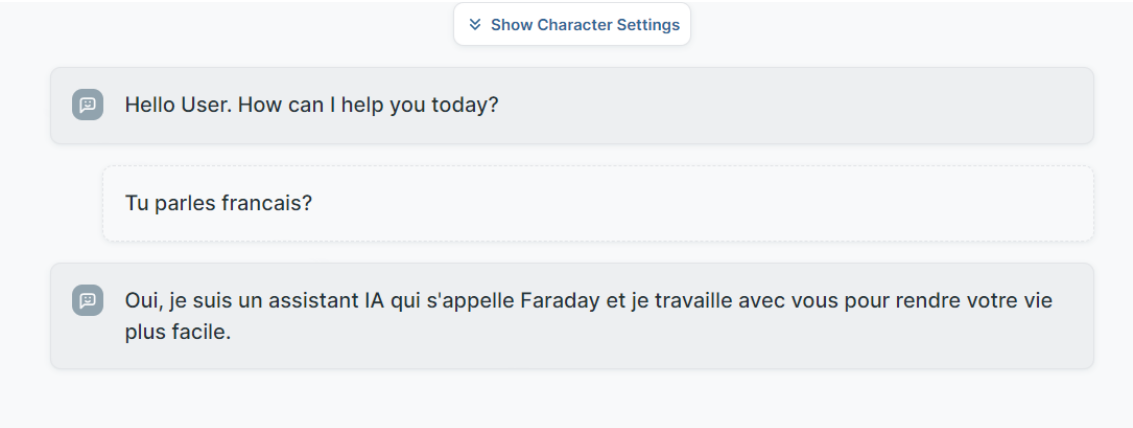

Figure 1: Exemple de conversation et interface Faraday

Mais dans leur dernière mise à jour ils ont rajouté la posibilité d'utiliser ces propres fichiers *.bin*. Nous avons testé cette fonctionalité avec le modèle gpt-4 Vicuna [6] et voici le résultat :

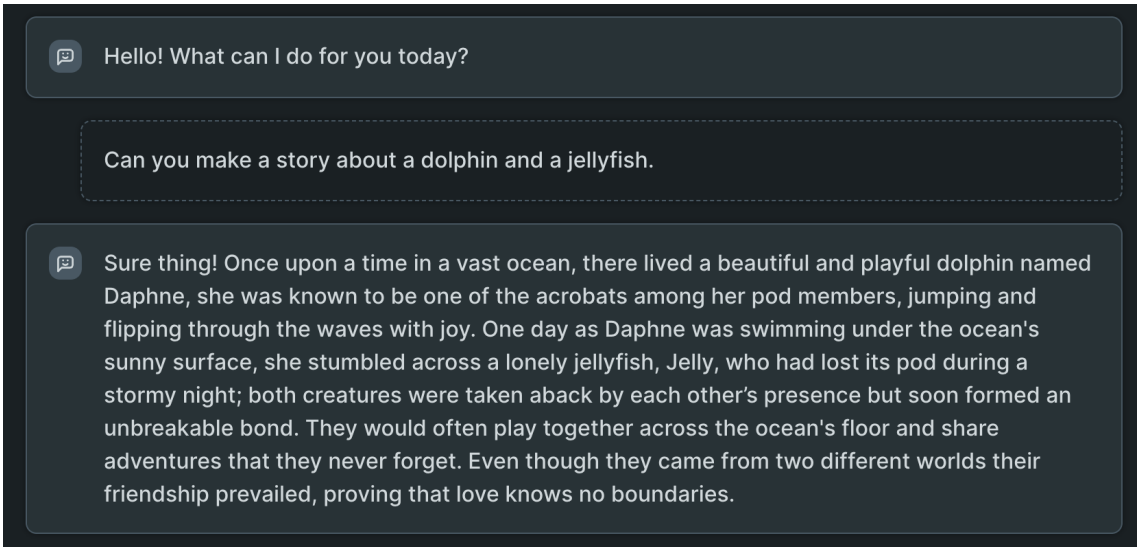

Figure 2: Exemple de conversation utilisant le .bin de GPT4 Vicuna

5 Autre

Nous avons également essayé Alpaca Lora [7] que vous pouvez essayer dans ce notebook Google Collab. Nous avons également testé le modèle Vigogne [8], qui est un modèle en français dans ce notebook Google Collab mis à disposition.

6 Conclusion

Faraday est l'option la plus simple, plus efficace et dotée d'une interface utilisateur développée, testée jusqu'à présent. En effet, il suffit de télécharger le client sur leur site et de choisir le modèle que l'on souhaite utiliser pour commencer à converser avec l'IA. Il n'y a pas besoin de toucher au code ou de taper des commandes dans un terminal, c'est même donc accessible aux non-informaticiens. Si l'on veut pousser plus loin et utiliser nos propres modèles téléchargés ou entrainés nous-même, nous avons la possibilité de le faire simplement en copiant un fichier *.bin* dans le dossier des modèles de Faraday.

On peut retrouver ici une liste [9] de différents modèles pré-entrainés avec les fichiers contenant les poids des neuronnes et de la documentation. Si cela ne suffit pas et que vous voulez vous penchez plus en détails dans la fabrication de modèles et l'entrainement de ceux-ci llama.cpp ou alpaca.cpp sont fais pour ça. Leurs github sont très détaillés mais leur utilisation demande des connaissances et de ne pas avoir peur de se "salir les mains". Enfin OpenLLama peut être une alternative à llama.cpp disponible en open source, mais il demande une plus grande puissance de calcul car la quantization n'est pas disponible.

References

[1] Large Language Model, URL: https://en.wikipedia.org/wiki/Large_language_model.

[2] LLaMA, Wikipédia, URL: https://en.wikipedia.org/wiki/LLaMA.

[3] openLLaMA, github, URL: https://github.com/openlm-research/open_llama.

[4] openLLaMA, huggingface, URL: https://huggingface.co/openlm-research/open_llama_3b/tree/main.

[5] Faraday, URL: https://faraday.dev/.

[6] Modèle GPT4 Vicuna, URL : https://huggingface.co/TheBloke/gpt4-x-vicuna-13B-GGML.

[7] Alpaca Lora, URL : https://github.com/tloen/alpaca-lora/.

[8] Vigogne, URL : https://huggingface.co/bofenghuang/vigogne-instruct-7b.

[9] Liste de modèles, URL : https://huggingface.co/TheBloke.