Projet NLP

Fine-tuning d'un modèle pré-entraîné (GPT2)

DA FONSECA Alexis & **ABDELLI** Khalid
& **BENSALAH** Lockman & **LOUOUNGOU** Michael

Étudiants en deuxième année de Master
Mathématiques et Applications à l'Université Gustave Eiffel

11 Avril 2025

TABLE DES MATIÈRES

INTRODUCTION

Les Large Language Models (LLM), ou modèles de langage de grande taille, représentent une avancée majeure dans le domaine de l'intelligence artificielle. Entraînés sur d'immenses volumes de données textuelles, ces modèles sont capables de comprendre, générer et résumer du langage naturel avec une finesse impressionnante.

Leur capacité à interagir de manière fluide avec les utilisateurs et à automatiser des tâches complexes ouvre de nouvelles perspectives pour les entreprises. Que ce soit pour l'assistance client, la génération de contenus, l'analyse de données non structurées ou encore l'aide à la prise de décision, les LLM s'imposent comme des outils stratégiques, permettant de gagner en efficacité, en réactivité et en innovation. Le temps et la puissance de calcul nécessaires lors de l'entraînement sont tellement importants qu'il est impossible pour nous, étudiants, de le réaliser sur nos ordinateurs personnels.

Ce document présente la démarche complète de fine-tuning du modèle GPT-2 avec la technique LoRA (Low-Rank Adaptation) sur le dataset *Alpaca*. Le but est d'adapter un modèle pré-entrainé à un jeu de données spécifique en optimisant uniquement une petite partie des paramètres.

CHAPITRE 1

OUTILS UTILISÉS

1.1 Dataset mis à notre disposition

L'entraînement d'un modèle de LLM nécessite en général une grande quantité de données textuelles. Cependant, dans le cadre du fine-tuning, on utilise un modèle déjà très entraîné. Il est donc possible d'obtenir de bonnes performances avec un volume de données bien plus restreint, à condition que celles-ci soient bien ciblées. C'est ce que nous mettons en pratique dans ce projet.

Nous avons choisi d'utiliser le jeu de données *Alpaca*, disponible sur Hugging Face via ce lien : `https://huggingface.co/datasets/tatsu-lab/alpaca`. Ce dataset contient plusieurs colonnes : `instruction`, `input`, `output`, et `text`.

1.1.1 Origine

Le dataset *Alpaca* a été créé par l'équipe de Stanford. Il s'inspire du dataset InstructGPT d'OpenAI et contient des paires *Instruction* - *Réponse*, parfois enrichies d'un contexte (*Input*).

1.1.2 Structure des données

Chaque exemple dans le dataset est structuré de la façon suivante :

— **instruction** : la tâche à accomplir.

— **input** : un contexte optionnel.

— **output** : la réponse attendue.

Ces exemples sont préparés sous forme de texte concaténé avec une structure typique :

```
Instruction: ...\nInput: ...\nResponse: ...
```

1.2 Le modèle GPT-2

1.2.1 Présentation générale

GPT-2 (Generative Pretrained Transformer 2) est un modèle de génération de texte développé par OpenAI. Il repose entièrement sur l'architecture Transformer, introduite par Vaswani et al. en 2017. Contrairement aux modèles bidirectionnels comme BERT, GPT-2 est un modèle unidirectionnel autorégressif, c'est-à-dire qu'il prédit chaque token à partir de ceux qui le précèdent uniquement.

Le modèle a été pré-entraîné sur un très grand corpus textuel de l'internet (WebText) via une tâche de langage classique consistant à prédire le prochain mot dans une séquence.

1.2.2 Architecture

GPT-2 repose sur la partie "decodeur" du Transformer. Chaque bloc de son architecture comprend :

— Une couche d'attention multi-tête (*multi-head self-attention*) avec masque causal ;

— Une normalisation de couche (*LayerNorm*) avant chaque sous-bloc ;

— Une couche feedforward positionnelle (MLP) composée de deux couches linéaires séparées par une activation ReLU ou GELU ;

— Des connexions résiduelles (*residual connections*) entre chaque sous-bloc.

1.2.3 Tokenisation

La tokenisation utilisée par GPT-2 repose sur le Byte Pair Encoding (BPE), qui segmente le texte en sous-unités fréquentes. Cette approche permet de gérer efficacement les mots rares ou inconnus et assure une représentation compacte et cohérente. Par exemple, un mot inconnu peut être découpé en préfixes et suffixes connus.

Le tokenizer de GPT-2 ne contient pas de token de padding (`[PAD]`) par défaut ; on utilise donc généralement le token de fin de séquence (`[EOS]`) pour cet usage dans le fine-tuning.

1.2.4 Masque d'attention causale

Pour garantir que le modèle ne puisse pas accéder aux tokens futurs lors de la génération, GPT-2 emploie un **masque d'attention causale**. Cela signifie que lors de l'entraînement comme lors de l'inférence, la position t ne peut accéder qu'aux positions $\leq t$. Ainsi, l'attention est strictement dirigée vers le passé (ou présent), empêchant toute fuite d'information depuis le futur.

1.2.5 Caractéristiques techniques

Les principales caractéristiques de la version de GPT-2 utilisée dans ce projet (la plus petite) sont :

— Nombre de couches (blocs Transformer) : 12

— Dimensions des vecteurs cachés : 768

— Nombre de têtes d'attention : 12

— Nombre total de paramètres : 124 millions

CHAPITRE 2

FINE-TUNING AVEC LORA

2.1 La méthode LoRA

2.1.1 Outils et bibliothèques utilisés

Le fine-tuning a été réalisé à l'aide de plusieurs bibliothèques incontournables dans le domaine du NLP :

- `transformers` : pour la gestion du modèle GPT-2 et de son tokenizer, ainsi que l'entraînement via le `Trainer`.
- `datasets` : pour charger et manipuler efficacement le dataset *Alpaca*.
- `peft` : pour appliquer le fine-tuning via la méthode LoRA.
- `accelerate` : pour faciliter l'entraînement sur GPU.
- `bitsandbytes` : pour supporter la quantification et réduire la mémoire utilisée lors de l'entraînement.

2.1.2 Principe de LoRA

LoRA, pour *Low-Rank Adaptation*, est une méthode efficace de fine-tuning qui vise à ne pas réentraîner tous les poids d'un grand modèle, mais uniquement une faible portion via des matrices d'adaptation. Elle repose sur le fait que les mises à jour nécessaires aux performances du modèle peuvent être bien approximées dans un sous-espace de faible rang.

Concrètement, au lieu de mettre à jour une matrice de poids $W \in \mathbb{R}^{d\times k}$ directement, LoRA introduit deux matrices entraînables $A \in \mathbb{R}^{d\times r}$ et $B \in \mathbb{R}^{r\times k}$ (où $r \ll d, k$), de sorte que :

$$W' = W + alpha \times AB$$

Cette stratégie permet de fortement réduire le nombre de paramètres à ajuster, tout en conservant la structure et les performances du modèle initial. Le facteur α permet de rééquilibrer l'échelle des matrices d'adaptation.

LoRA est particulièrement adaptée aux modèles de grande taille comme GPT-2, car elle permet un entraînement efficace même avec peu de ressources.

2.2 Apprentissage du modèle avec LoRA

2.2.1 Objectif du modèle de langage causal

Le modèle GPT-2 est un *Language Model* causal, dont l'objectif est de prédire la probabilité du prochain token x_t connaissant les précédents $(x_1, \ldots, x_{t-1})$. L'entraînement cherche donc à maximiser la vraisemblance :

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \sum_{t=1}^{T_i} \log P_\theta(x_t^{(i)} \mid x_1^{(i)}, \ldots, x_{t-1}^{(i)})$$

Ce problème se reformule comme la minimisation de la perte par *cross-entropie* :

$$\mathcal{L}_{\text{CE}} = -\sum_{t=1}^{T} y_t \cdot \log(\hat{y}_t)$$

où $\hat{y}_t$ est la distribution prédite par le modèle (via softmax), et y_t la vérité terrain (représentée en one-hot).

2.2.2 Fine-tuning classique vs LoRA

Dans un fine-tuning classique, les poids θ du modèle entier sont mis à jour par descente de gradient :

$$\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}_{\text{CE}}$$

Cependant, LoRA (*Low-Rank Adaptation*) modifie cette approche en ajoutant des matrices de bas rang $A \in \mathbb{R}^{d \times r}$ et $B \in \mathbb{R}^{r \times d}$ à certaines couches du modèle, tout en gardant les poids originaux gelés. L'opération linéaire devient :

$$h = (W + \alpha BA)x$$

où :

— W est la matrice de poids originale (gelée),

— A, B sont les matrices apprises pendant l'entraînement,

— α est un facteur de mise à l'échelle ($\alpha = \texttt{lora\_alpha}/r$).

Seuls les paramètres A et B sont mis à jour :

$$A \leftarrow A - \eta \cdot \nabla_A \mathcal{L}_{\text{CE}}, \quad B \leftarrow B - \eta \cdot \nabla_B \mathcal{L}_{\text{CE}}$$

2.2.3 Optimisation et entraînement

L'optimisation est effectuée avec l'algorithme AdamW sur les paramètres A et B uniquement. Le reste du modèle reste figé.

L'entraînement est réalisé sur un jeu de données tokenisé, avec accumulation de gradients (`gradient_accumulation_steps`) et calculs en précision mixte (FP16) pour accélérer les calculs.

La fonction de perte est agrégée sur les séquences tokenisées, puis utilisée pour mettre à jour uniquement les modules ciblés par LoRA (ici, les couches d'attention nommées `c_attn` dans GPT-2).

2.2.4 Implémentation

Dans notre projet, LoRA a été appliquée sur les modules du modèle GPT-2. Ce module est responsable de la projection linéaire des entrées dans les espaces des requêtes, clés et valeurs dans le mécanisme d'attention. Les hyperparamètres choisis sont :

— **Rang (r)** : 8

— **Alpha** : 32

— **Dropout LoRA** : 0.1

— **Bias** : aucun

Ces valeurs ont été choisies en s'inspirant des réglages standards recommandés dans les travaux initiaux sur LoRA, notamment pour des modèles de taille similaire à GPT-2. Un rang $r = 8$ permet de capturer des variations pertinentes tout en limitant le nombre de paramètres ajoutés. Le coefficient $\alpha = 32$ permet de rééquilibrer la contribution des matrices LoRA par rapport aux poids d'origine. Un dropout de 0.1 est utilisé pour régulariser l'entraînement et éviter le surapprentissage, en particulier avec un nombre d'exemples relativement restreint. Ce compromis entre efficacité mémoire, capacité d'adaptation, et stabilité a été validé empiriquement dans de nombreux projets LoRA sur des modèles de langage.

CHAPITRE 3

EVALUATION DU MODÈLE

3.1 Comparaison entre deux modèles de génération de texte

3.1.1 Interface graphique et gestion du prompt

Ce projet vise à comparer deux modèles de génération de texte : un modèle de base (`GPT-2`) et une version fine-tunée via la méthode `LoRA` (Low-Rank Adaptation). Pour faciliter l'expérimentation et l'interaction utilisateur, une interface graphique a été développée à l'aide de la bibliothèque `Streamlit`, qui permet de créer des applications web en Python avec un minimum de code.

L'interface présente une zone de saisie (`st.text_area`) où l'utilisateur peut entrer une instruction en langage naturel. Ce prompt est ensuite automatiquement encapsulé dans un format structuré :

```
### Instruction:
[texte de l'utilisateur]

### Response:
```

Ce format est souvent utilisé dans le fine-tuning de modèles de type instruction-based (comme Alpaca, Vicuna, ou Falcon-Instruct) afin de mieux contextualiser les requêtes.

Les paramètres utilisés lors de la génération sont :

— `do_sample=True` : active l'échantillonnage aléatoire,

— `temperature=0.8` : régule la créativité du modèle,

— `top_p=0.95` : applique la stratégie de *nucleus sampling*,

— `top_k=50` : limite aux 50 tokens les plus probables,

— `max_length=100` : borne la réponse à 100 tokens maximum.

Une fois les réponses générées, elles sont affichées à l'utilisateur via deux encadrés distincts :

— `st.success()` (encadré vert) pour la réponse du modèle **LoRA**,

— `st.info()` (encadré bleu) pour la réponse du modèle **GPT-2**.

3.1.2 Comparaison qualitative entre les modèles

Le but de cette interface est de mettre en évidence les différences entre un modèle de langage généraliste (GPT-2) et un modèle adapté à une tâche spécifique via la méthode `LoRA`.

Le modèle GPT-2, est moins performant sur des tâches qui nécessisent de restituer connaissances sur des sujets précis. Par exemple sur les questions conçernanat des guides touristiques, GPT-2 donnera une réponse très vague, là où le modèle LoRA donnera des réponses précises en donnant des lieux de villes à visiter par exemple.

Le modèle LoRA semble être moins bon en logique, par exemple il ne considère que le café est un soda, là où GPT-2 sait faire cette différence.

En résumé, le modèle LoRA répond mieux aux questions posées en donnant des réponses plus ciblées par rapport à la question là où GPT-2 n'hésite pas à "halluciner",c'est-à-dire donner une réponse fausse, et répond de manière beaucoup plus générale. Les deux modèles font toutefois pas mal d'erreurs, ces erreurs peuvent être dues à un overfitting sur les données d'entraînement.

CHAPITRE 4

CONCLUSION ET LIMITES DE NOTRE ALGORITHME

Le fine-tuning de GPT-2 avec la méthode LoRA appliquée au dataset *Alpaca* a permis de transformer un modèle généraliste en un assistant spécialisé dans le traitement d'instructions. Grâce à l'architecture légère de LoRA, nous avons pu affiner le comportement du modèle sans nécessiter un grand volume de ressources, ni modifier l'intégralité de ses poids d'origine.

Le modèle ainsi obtenu est capable de répondre de manière structurée à une variété d'instructions en anglais, dans un format *Instruction* → *Input* → *Response*. Il présente des performances satisfaisantes pour des tâches comme la reformulation, l'explication, la génération de courts textes ou la réponse à des questions simples. Ces capacités ont été acquises grâce à la diversité et à la clarté du dataset *Alpaca*.

Capacités du modèle fine-tuné

- **Explication** : Capable d'expliquer des concepts ou des différences.
- **Résumé** : Peut condenser un texte en une phrase.
- **Complétion / Génération** : Génère des réponses ou poursuit une instruction.
- **Classification simple** : Classe des éléments selon un critère donné.
- **Q/R simples** : Répond à des questions factuelles.

Limites du modèle fine-tuné

- **Traduction** : Le modèle n'a pas été entraîné sur des exemples de traduction et ne répond pas de manière fiable.
- **Multilinguisme** : Ne comprend que l'anglais, car le dataset est exclusivement anglophone.
- **Calculs ou raisonnement complexe** : Ne sait pas résoudre des problèmes mathématiques ou logiques avancés.
- **Instructions ambiguës ou vagues** : Génère des réponses incohérentes si le prompt est mal défini.
- **Dialogue multi-tours** : GPT-2 n'a pas de mémoire de contexte au-delà de son entrée immédiate.

En conclusion, le fine-tuning de GPT-2 avec LoRA et Alpaca démontre que l'on peut, avec peu de ressources, spécialiser un grand modèle de langage pour accomplir efficacement un ensemble de tâches ciblées. Toutefois, ses performances restent limitées par la nature du dataset d'entraînement et par les capacités initiales de GPT-2, notamment en ce qui concerne la compréhension multilingue, la traduction, ou le raisonnement avancé.