

Mathias **BEAUREPERE**

Alex **SCHLAGETER**



# Rapport :

## Recherches sur les LLM



**UE** : Projet de fin d'année

**Responsable** : M. FORESTIER

**2023 - 2024**



# Sommaire

<b>Introduction.....</b>	<b>5</b>
<b>1 Recherche Initiale et Mise en Place des Outils.....</b>	<b>8</b>
1.1 Recherche Préliminaires.....	8
1.2 Installation et Configuration des Outils.....	8
<b>2 Exploration et Test des Modèles.....</b>	<b>12</b>
2.1 Modèles Téléchargés.....	12
2.2 Résultats des Tests et Observations.....	13
2.3 Observations Générales.....	16
<b>3 Développement de l'Application de RAG.....</b>	<b>17</b>
<b>4 Mise en Place des Templates de Prompts.....</b>	<b>20</b>
4.1 Création et Utilisation des Templates.....	20
4.2 Exemples de Templates Simples.....	22
4.3 Avantages des Templates.....	23
<b>5 Fine-Tuning des Modèles.....</b>	<b>25</b>
5.1 Recherche et Exploration du Fine-Tuning.....	25
5.2 Utilisation de Google Colab pour le Fine-Tuning sur un Dataset en Anglais (Alpaca).....	25
5.3 Défis Techniques et Résolutions.....	27
<b>6 Problèmes et Solutions lors du Fine-Tuning.....</b>	<b>28</b>
6.1 Entraînement Spécifique et Création de Dataset.....	28
6.2 Utilisation et Transformation des LoRAs.....	28
6.3 Problèmes Résolus et Nouveaux Défis.....	29
6.4 Explications des Concepts Techniques.....	30
<b>7 Extension Chrome Modifiée : Ollama-Reply.....</b>	<b>31</b>
7.1 Contexte et Objectif.....	31
7.2 Description de l'Extension.....	31
7.3 Processus de Modification.....	31
7.4 Résultats et Observations.....	33
7.5 Conclusion.....	34
<b>8 Documentation et Instructions.....</b>	<b>35</b>
8.1 Création de Guides pour la Création Rapide de Modelfiles.....	35
8.2 Amélioration Continue de la Compréhension des Modelfiles.....	35
8.3 Mise à Jour des Guides en Fonction des Évolutions des Outils et des Techniques.....	36
<b>9 Les LLM : Une Technologie en Évolution Constante.....</b>	<b>38</b>

9.1 Défis de la Documentation et des Tutoriels.....	38
9.2 Évolution Technologique Continue.....	39
<b>10 Conclusion.....</b>	<b>40</b>
Ressenti des Participants.....	41
<b>Table des illustrations.....</b>	<b>44</b>
<b>Annexes.....</b>	<b>45</b>

# Introduction

Ce projet se concentre sur la personnalisation des modèles de langage (LLM) disponibles en open-weights (open-source), afin d'optimiser leurs performances pour des applications spécifiques. Les modèles de langage, ou LLM, sont des outils puissants capables de générer du texte, de répondre à des questions et de réaliser diverses tâches linguistiques en se basant sur des données d'entraînement. Cependant, pour répondre à des besoins spécifiques, ces modèles doivent souvent être personnalisés. Notre projet vise à explorer et à mettre en œuvre différentes méthodes de personnalisation des LLM afin d'optimiser leurs performances pour des applications spécifiques.

Les **objectifs principaux** de ce projet sont :

- **Personnaliser** les modèles de langage disponibles en open-weights : Adapter les LLM pour des tâches spécifiques afin d'améliorer leur efficacité et leur pertinence. Les open-weights désignent les modèles dont les paramètres peuvent être ajustés librement par les utilisateurs.
- **Utiliser Ollama et Open WebUI** : Mettre en œuvre la personnalisation en utilisant des outils accessibles et performants comme Ollama et Open WebUI.

Les modèles de langage jouent un rôle crucial dans de nombreuses applications modernes, allant de la génération de texte à la compréhension du langage naturel. Leur capacité à apprendre et à s'adapter à de nouvelles données en fait des outils essentiels pour l'intelligence artificielle (IA) et le traitement du langage naturel (NLP). Cependant, l'efficacité de ces modèles peut être limitée lorsqu'ils sont appliqués à des domaines ou à des tâches spécifiques sans personnalisation. D'où l'importance des méthodes de personnalisation, qui permettent d'affiner les capacités des LLM pour mieux répondre aux exigences particulières des utilisateurs.

Pour atteindre nos objectifs, nous avons utilisé deux outils principaux : Ollama et Open WebUI.

- **Ollama** : Ollama est une plateforme qui facilite le déploiement rapide des modèles de langage. Elle offre une infrastructure robuste pour l'hébergement et la gestion des LLM, ce qui nous permet de concentrer nos efforts sur la personnalisation des modèles plutôt que sur les aspects techniques du déploiement.
- **Open WebUI** : Open WebUI est une interface utilisateur web qui permet de tester et d'interagir avec les modèles de langage de manière intuitive. En utilisant Docker Desktop, une application qui facilite la création et le déploiement d'applications dans des conteneurs, Open WebUI peut être rapidement déployé, offrant ainsi un environnement flexible pour expérimenter avec les prompts et les personnalisations des modèles.

Pour personnaliser les modèles de langage, nous avons exploré trois niveaux principaux de personnalisation :

1. **Prompts spécifiques** : Cette méthode consiste à créer des prompts ou des questions spécifiques qui dirigent le modèle vers des réponses précises. Un prompt est une instruction ou une question donnée au modèle pour générer une réponse. En utilisant des prompts bien conçus, nous pouvons exploiter les capacités des LLM pour générer des réponses adaptées à des contextes particuliers.

**Exemple** : Pour un service client, en formulant des questions spécifiques comme "Comment puis-je réinitialiser mon mot de passe sur l'application mobile ?", le modèle peut fournir des réponses précises et utiles.

2. **RAG (Retrieval-Augmented Generation)** : Le RAG est une technique qui combine la génération de texte avec la récupération d'informations. En fournissant au modèle une base de connaissances sous forme de

documents, nous augmentons ses capacités à répondre de manière informée et contextuelle. La récupération d'informations permet au modèle d'accéder à des données externes pour enrichir ses réponses.

**Exemple** : En intégrant des documents spécifiques à un domaine médical, le modèle peut utiliser ces informations comme "pré-connaissance" pour fournir des réponses médicales précises.

3. **Fine-tuning des modèles** : Le fine-tuning implique de réentraîner un modèle de langage sur un dataset spécialisé pour modifier ses poids (les paramètres internes du modèle) et l'adapter à des tâches spécifiques. Cette méthode est plus complexe et nécessite une puissance de calcul significative, mais elle permet d'obtenir des modèles hautement spécialisés.

**Exemple** : En réentraînant un modèle avec des données spécifiques à une entreprise ou à un domaine scientifique, nous pouvons améliorer ses capacités à répondre avec une grande précision dans ce contexte particulier.

# 1 Recherche Initiale et Mise en Place des Outils

## 1.1 Recherche Préliminaires

Pour initier notre projet, nous avons entrepris une phase de recherche approfondie pour comprendre les concepts de base et les meilleures pratiques concernant les modèles de langage (LLM) et leur personnalisation. Cette recherche préliminaire a inclus plusieurs étapes clés :

- **Visionnage de vidéos éducatives** : Nous avons regardé de nombreuses vidéos sur des plateformes telles que YouTube et des cours en ligne pour comprendre les fondements des LLM, le RAG (Retrieval-Augmented Generation), et le fine-tuning des modèles. Ces vidéos nous ont fourni une base solide de connaissances théoriques et pratiques.
- **Lecture d'articles et de documents de recherche** : Nous avons étudié des articles scientifiques et des documents techniques pour approfondir notre compréhension des méthodes de personnalisation des LLM. Ces lectures nous ont permis de découvrir les dernières avancées et les meilleures pratiques dans le domaine.
- **Consultation de la documentation officielle** : Pour une compréhension détaillée des outils que nous allions utiliser, nous avons consulté la documentation officielle d'Ollama et d'Open WebUI. Cette documentation nous a aidé à comprendre les fonctionnalités, les configurations possibles et les étapes nécessaires pour l'installation et l'utilisation de ces outils.

Cette démarche de recherche préliminaire nous a permis de bien comprendre les concepts clés et de préparer efficacement la phase de mise en place des outils.

## 1.2 Installation et Configuration des Outils



Une fois que nous avons une bonne compréhension théorique, nous sommes passés à l'installation et à la configuration des outils nécessaires pour notre projet.

### 1.2.1 Installation d'Ollama

Ollama est une plateforme qui facilite le déploiement et la gestion des modèles de langage. Pour l'installer, nous avons suivi les étapes suivantes :

1. **Téléchargement et installation** : Nous avons téléchargé Ollama depuis le [site officiel](#) et suivi les instructions pour l'installation sur notre système.
2. **Configuration initiale** : Une fois installé, nous avons configuré Ollama en fonction des besoins spécifiques de notre projet (pull des modèles que nous souhaitons utiliser).
3. **Tests de fonctionnement** : Pour nous assurer que tout était correctement configuré, nous avons effectué des tests de base pour vérifier que les modèles de langage pouvaient être chargés et interrogés correctement.

### 1.2.2 Installation et configuration d'Open WebUI avec Docker Desktop

1. **Installation de Docker Desktop** : Docker Desktop est un outil qui facilite le déploiement d'applications dans des conteneurs. Nous avons commencé par installer Docker Desktop sur notre machine en suivant les instructions fournies sur [le site officiel de Docker](#).
2. **Configuration du conteneur Docker** : En utilisant Docker, nous avons configuré le conteneur nécessaire pour exécuter Open WebUI à l'aide d'une simple commande ([voir la doc](#)).
3. **Déploiement et tests** : Après avoir configuré les conteneurs, nous avons déployé Open WebUI et effectué des tests pour nous assurer que l'interface fonctionnait correctement. Nous avons vérifié que les modèles pouvaient être chargés, interrogés et que les résultats étaient affichés correctement.

### 1.2.3 Tentatives d'optimisation avec CUDA

Pour améliorer les performances des modèles de langage, en particulier lors de l'inférence, nous avons exploré l'utilisation de CUDA (Compute Unified Device Architecture). CUDA est une plateforme de calcul parallèle développée par NVIDIA qui permet d'accélérer les calculs en utilisant les GPU.

- **Installation des pilotes CUDA** : Nous avons tenté d'installer les pilotes CUDA nécessaires sur notre machine. Cela incluait l'installation des pilotes GPU compatibles et la configuration de l'environnement CUDA. Cependant, nous avons rencontré des difficultés techniques lors de cette étape, notamment des problèmes de compatibilité entre les versions des pilotes et notre matériel.
- **Intégration avec Docker** : Nous ne savions pas initialement si les conteneurs Docker devaient être configurés pour utiliser CUDA. Nous avons essayé de modifier nos configurations Docker pour tirer parti de CUDA, en nous assurant que les conteneurs Docker pouvaient accéder aux GPU pour l'accélération des calculs. Malgré plusieurs tentatives, nous n'avons pas réussi à faire fonctionner correctement CUDA avec Docker, les conteneurs ne parvenant pas à reconnaître les GPU disponibles.
- **Tests de performance** : En raison des problèmes rencontrés lors de l'installation et de l'intégration, nous n'avons pas pu effectuer des tests de performance complets. Les essais initiaux n'ont pas montré d'amélioration significative.

Avec le recul, maintenant que le projet est terminé, nous avons réalisé que cette configuration n'était pas nécessaire car c'est l'application Ollama qui effectuait tous les calculs, et non les conteneurs Docker. Ces tentatives d'optimisation avec CUDA n'ont pas abouti comme prévu, mais elles nous ont permis d'approfondir notre compréhension des exigences matérielles et des configurations nécessaires pour

l'accélération GPU. Nous avons documenté les problèmes rencontrés et les solutions envisagées pour de futures tentatives d'intégration.

## 2 Exploration et Test des Modèles

### 2.1 Modèles Téléchargés

Pour notre projet, nous avons choisi de travailler avec plusieurs modèles disponibles sur la plateforme Ollama. Nous avons principalement exploré les **modèles [llama3](#)** dans leurs variantes 7b et 70b. Il s'agit des versions les plus avancées de la série des modèles Llama développées par [Meta](#). Ces modèles ont été sélectionnés en raison de leur performance reconnue et de leur adaptabilité aux différentes tâches de traitement du langage naturel (NLP).

1. **Llama3 7b** : Ce modèle, avec ses 7 milliards de paramètres, est suffisamment léger pour être exécuté sur des machines avec des ressources limitées tout en offrant une performance décente pour de nombreuses tâches NLP.
2. **Llama3 70b** : Avec 70 milliards de paramètres, ce modèle est beaucoup plus puissant et capable de générer des réponses plus précises et contextuellement pertinentes. Cependant, il nécessite des ressources matérielles considérablement plus importantes.

Nous avons également testé d'**autres modèles** pour évaluer leur performance en comparaison avec llama3 :

3. **[Phi-3](#) (3.8b)** : Le modèle de Microsoft qui succède donc à Phi-2. Il se distingue par sa grande capacité de généralisation dans divers domaines et par ses améliorations en termes d'efficacité énergétique et de vitesse.
4. **[Mistral](#) (7.2b)** : Un modèle de langage développé par une entreprise française du même nom, axé sur l'optimisation et la performance.
5. **[Gemma](#) (8.5b)** : Gemma a été développé par Google et s'inspire donc de [Gemini](#). Il a été entraîné sur un ensemble de données très diversifié et avec

l'utilisation de diverses techniques de nettoyage et de filtrage des données pour s'assurer de sa sécurité.

6. **Qwen (4b)** : Le modèle conçu par [Alibaba Cloud](#).

## 2.2 Résultats des Tests et Observations

Pour être sûr que l'utilisation des modèles llama3 était effectivement le bon choix, nous avons très rapidement effectué des tests très basiques notamment avec un prompt permettant de répondre à un mail spécifique à l'aide d'une brève piste de réponse (Voir [Annexe I](#)). Pour chacun des modèles que nous voulions tester, nous avons fourni exactement le même prompt avec un même mail d'exemple et les mêmes pistes de réponses (Voir [Annexe II](#)). Les résultats de ces tests pour chacun des modèles est disponible dans l'[Annexe III](#).

Voici les différentes observations que l'on peut faire de ces résultats, classées du pire modèle au meilleur en fonction des performances :

- **Qwen (4b)** : Il s'agit sans aucun doute des pires résultats obtenus. Le modèle n'a même pas compris la demande, répondant complètement à côté et négligeant même les petits détails tels que la signature et la formule de politesse avec le nom de l'interlocuteur. Cette performance s'explique aisément, le modèle étant principalement destiné au marché asiatique et étant quasiment uniquement entraîné sur de l'anglais et du chinois. De plus, il fait partie des plus petits modèles de la liste.
- **Mistral (7.2b)** : Contrairement à Qwen, Mistral a compris à qui il devait adresser son mail. Cependant, plusieurs problèmes sont constatés. Tout d'abord, la signature est dupliquée sans raison valable. Ensuite, l'objet du mail est en anglais, et pire encore, Mistral a inventé des projets passés de l'entreprise qui ne figurent jamais dans les pistes de réponse, ce qui rend ce

modèle difficile à utiliser sans modifications du prompt initial pour ajouter de la sécurité.

- **Gemma (8.5b)** : C'était le modèle le plus "lourd" à notre disposition (avec Llama 70b). Bien que sa réponse ne soit pas mauvaise, quelques points noirs sont à noter. Tout d'abord, son expression en français puis la signature incomplète. Malgré cela, le modèle s'en sort plutôt bien dans l'ensemble.
- **Phi-3 (3.8b)** : Pour un modèle aussi léger, le résultat est assez bluffant. Le français est impeccable, le modèle n'invente rien et va à l'essentiel. Seul petit point à noter, le modèle ne prend pas complètement en compte la signature donnée dans le prompt en utilisant des variables ([Your Name]...). Nous ne considérons pas cela comme un problème puisque nous n'indiquons pas clairement d'utiliser la signature d'exemple du prompt dans les mails suivants.
- **Llama3 (7b)** : Rien à redire sur le modèle. Il fait dans la simplicité mais n'omet pas les petits détails qui rendent le mail le plus professionnel possible. De plus, la signature est exactement mise sous la même forme que dans le prompt initial et est précédée de la formule de politesse finale suivie du prénom et nom comme demandé.

Évidemment, nous ne nous basions pas entièrement sur ces résultats pour choisir notre modèle principal, mais également sur les comparateurs en ligne et l'avis général des autres développeurs. Il est également important de prendre en compte la variabilité des réponses. Par exemple, en posant plusieurs fois l'exemple des mails, nous avons constaté parfois de très bonnes réponses de Mistral. Cependant, nous avons également remarqué une certaine régularité chez Llama3 ce qui nous a mené à le privilégier !

Enfin, nous avons mené une série de tests pour évaluer les performances des modèles Llama3 plus en détail, en utilisant notre application de RAG et de prompt

développée en Python. Les tests visaient à mesurer la pertinence des réponses générées, la vitesse de traitement et la consommation des ressources.

### 1. Llama3 7b :

- **Pertinence des Réponses** : Les réponses générées par le modèle 7b étaient globalement pertinentes, bien que parfois limitées en profondeur et en précision comparativement au modèle 70b.
- **Vitesse de Traitement** : Le modèle 7b offrait des temps de réponse acceptables, rendant l'application réactive même sur des machines avec des capacités de calcul modérées.
- **Consommation des Ressources** : La consommation de mémoire et de CPU était raisonnable, permettant une exécution fluide sans nécessiter de matériel haut de gamme.

### 2. Llama3 70b :

- **Pertinence des Réponses** : Les réponses étaient nettement plus précises et contextuellement adaptées, démontrant la capacité du modèle à gérer des requêtes complexes avec une grande exactitude.
- **Vitesse de Traitement** : En raison de sa taille, le modèle 70b avait des temps de réponse beaucoup plus longs, ce qui peut être un inconvénient pour des applications nécessitant une réponse rapide.
- **Consommation des Ressources** : La demande en ressources était très élevée, rendant difficile l'exécution du modèle sur des machines standards sans matériel spécialisé (GPU haute performance, grande quantité de RAM).

## 2.3 Observations Générales

**Comparaison des Performances :** Bien que certains modèles comme Phi-3, et Gemma aient montré des résultats corrects, à l'utilisation, on remarquait une différence notable dans les réponses. Les modèles llama3 7b et 70b se sont avérés être les plus adaptés à nos besoins, offrant un bon équilibre entre la qualité des réponses et l'efficacité des ressources. En pratique, llama3 7b s'est révélé être le plus confortable à utiliser contrairement à llama3 70b qui fonctionnait beaucoup trop lentement sur nos machines.



### 3 Développement de l'Application de RAG

Pour le développement de l'application de RAG, nous avons choisi Python en raison de sa flexibilité et de la richesse de ses bibliothèques pour le traitement du langage naturel (NLP). Notre objectif était donc de créer une application de RAG (Retrieval-Augmented Generation) et de prompt qui fonctionne en ligne de commande, sans interface graphique initialement.

1. **Conception de l'architecture** : Nous avons commencé par définir l'architecture de l'application. Celle-ci devait être modulaire pour permettre des mises à jour et des améliorations faciles. **L'application comprenait plusieurs modules principaux** :
  - **Module de Récupération de Documents** : Ce module permet de récupérer les documents à partir d'une base de connaissances pour les fournir au modèle comme contexte.
  - **Module de Génération de Réponses** : Ce module utilise le modèle de langage pour générer des réponses basées sur les prompts spécifiques et les documents récupérés.
  - **Module de Gestion des Prompts** : Ce module gère les prompts spécifiques pour diriger le modèle vers des réponses précises.
2. **Développement de l'application** : Nous avons implémenté les différents modules en Python, en utilisant des bibliothèques telles que `transformers` pour le traitement du langage naturel et `requests` pour les interactions avec les API. L'application en ligne de commande permet à l'utilisateur de spécifier les prompts et les documents, et de recevoir des réponses générées par le modèle.
  - **app.py** : Le fichier principal de l'application qui orchestre les différentes opérations. Il initialise les composants, gère les entrées utilisateur et affiche les réponses générées.

- **config.py** : Ce fichier contient les configurations nécessaires pour l'application, telles que les chemins des fichiers de données et les paramètres des modèles.
  - **get\_embedding\_function.py** : Ce script fournit des fonctions pour obtenir les embeddings des documents, essentiels pour le RAG.
  - **populate\_database.py** : Utilisé pour remplir la base de données avec les documents à partir desquels les réponses seront générées.
  - **query\_data.py** : Contient les fonctions nécessaires pour interroger la base de données et récupérer les documents pertinents.
  - **test\_rag.py** : Un script de test pour vérifier que les fonctionnalités de récupération et de génération fonctionnent correctement.
3. **Base de données Chroma** : Nous utilisons Chroma, qui est une base de données vectorielle. Une base de données vectorielle stocke des "embeddings", qui sont des représentations numériques des documents. Ces embeddings permettent de comparer les documents de manière efficace.

#### **Voici ce que nous faisons :**

1. **Utilisation d'un modèle d'embeddings** disponible sur Ollama : Nous transformons les documents (comme des fichiers PDF) en embeddings en utilisant un modèle d'embeddings disponible sur Ollama. Ces embeddings capturent le contenu des documents sous une forme numérique que le modèle de langage peut comprendre.
2. **Stockage dans Chroma** : Une fois que nous avons ces embeddings, nous les stockons dans la base de données Chroma. Chroma nous permet de rechercher rapidement et efficacement parmi ces embeddings.
3. **Récupération des Documents** : Lorsqu'un utilisateur pose une question, l'application utilise Chroma pour trouver les documents les plus pertinents en comparant les embeddings de la question et des documents. Cela permet de

récupérer les documents qui contiennent les informations nécessaires pour répondre à la question.

En résumé, nous transformons les documents PDF en embeddings en utilisant un modèle d'Ollama, puis nous stockons ces embeddings dans Chroma pour pouvoir récupérer rapidement les documents pertinents lorsqu'une question est posée ([EmbedDB](#)) ([GitHub](#)) ([Chroma Docs](#)).

## 4 Mise en Place des Templates de Prompts

### 4.1 Création et Utilisation des Templates

#### 4.1.1 Création des Templates

La création de templates de prompts est une étape cruciale pour structurer les interactions avec les modèles de langage et obtenir des réponses précises et pertinentes. Les templates de prompts servent de guides pour formuler les questions et diriger les modèles vers des réponses spécifiques. Dans notre projet, nous avons développé des templates adaptés à diverses applications.

##### 1. Analyse des Besoins :

- **Méthodologie** : Nous avons procédé à une recherche en ligne pour identifier les types de questions fréquemment posées ainsi que les formats de réponses attendus. Cette démarche nous a permis de déterminer les variables dynamiques nécessaires pour chaque modèle.

##### 2. Conception des Templates :

- **Définition des Éléments Clés** : Chaque template comprend des éléments tels que le contexte, les instructions spécifiques et les variables dynamiques. Par exemple, pour la génération d'exercices de mathématiques, le template inclut le type de problème, la difficulté, et les étapes de solution.
- **Format et Structure** : Les templates sont créés en utilisant des placeholders (variables dynamiques) qui sont remplis au moment de l'exécution.

```
python Copier le code  
  
from langchain import PromptTemplate  
  
template = PromptTemplate("Please write a {adjective} sentence using a {noun}.")  
formatted_prompt = template.format(adjective="colorful", noun="flower")
```

Illustration 1 : Template de prompt avec [LangChain](#)

### 4.1.2 Utilisation des Templates dans Open Web UI

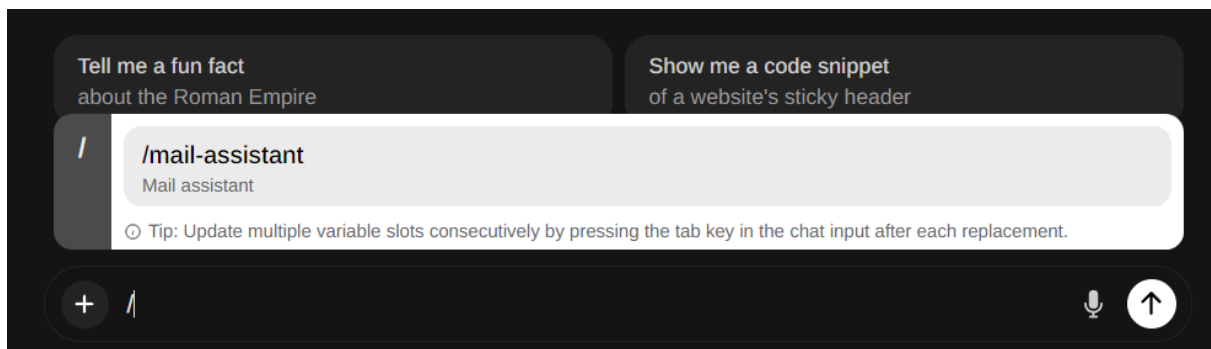
Nous avons intégré nos templates de prompts dans Open Web UI pour faciliter leur utilisation et permettre une interaction fluide avec les modèles de langage (Exemple dans l'[Annexe I](#)).

#### 1. Intégration des Templates :

- Configuration : Les templates sont ajoutés dans Open Web UI en les configurant dans les fichiers de paramètres de l'interface. Cela inclut la définition des templates de prompts et leur association à des fonctionnalités spécifiques de l'application. Par exemple, un template pour la recherche dans des articles scientifiques peut être configuré pour inclure des variables dynamiques telles que le domaine de recherche et les mots-clés.

#### 2. Interface Utilisateur :

- Sélection et Utilisation : Open Web UI permet aux utilisateurs de sélectionner et d'utiliser les templates en utilisant les commandes via `/`. Les utilisateurs peuvent choisir le template approprié, saisir les variables dynamiques requises, et envoyer la requête au modèle de langage.



*Illustration 2 : Utilisation des templates dans Open WebUI*

### 3. Tests et Validation :

- Nous avons brièvement effectué des tests pour vérifier la fonctionnalité de chaque template.

## 4.2 Exemples de Templates Simples

### 1. Template pour la **Génération d'Exercices de Mathématiques** :

- **Contexte** : "Générer un exercice de mathématiques pour un élève de niveau secondaire."
- **Instructions** : "Créer un problème d'algèbre de niveau intermédiaire et fournir les étapes de solution détaillées."
- **Variables Dynamiques** : Type de problème (algèbre, géométrie, etc.), Niveau de difficulté (facile, intermédiaire, difficile).

### 2. Template pour la **Recherche dans des Articles Scientifiques** :

- **Contexte** : "Effectuer une recherche dans une base de données d'articles scientifiques."

- **Instructions** : "Trouver les articles les plus récents sur le sujet de l'intelligence artificielle dans l'éducation."
- **Variables Dynamiques** : Domaine de recherche (IA, éducation, etc.), Mots-clés spécifiques, Type de résultat attendu (résumé, citation, etc.).

### 3. Template pour la **Réponse aux Emails** :

- **Contexte** : "Aider à rédiger une réponse à un email professionnel."
- **Instructions** : "Rédiger une réponse polie et formelle à un email de demande d'information sur un projet en cours."
- **Variables Dynamiques** : Nom du destinataire, Sujet de l'email, Informations spécifiques à inclure.

## 4.3 Avantages des Templates

### 1. Standardisation :

Les templates permettent de standardiser les requêtes, assurant ainsi une cohérence dans les réponses générées par les modèles de langage.

### 2. Efficacité :

Les templates simplifient le processus de formulation des questions, rendant l'interaction avec les modèles plus rapide et plus intuitive.

### 3. Précision :

En structurant les prompts, les templates aident à diriger les modèles vers des réponses plus précises et pertinentes, améliorant ainsi la qualité des résultats.

La mise en place des templates de prompts dans Open Web UI a considérablement amélioré notre capacité à interagir efficacement avec les modèles de langage. Les templates nous ont permis de structurer les requêtes et d'obtenir des réponses pertinentes adaptées à diverses applications.

Pour plus d'informations sur l'utilisation et la personnalisation des templates dans Open Web UI, vous pouvez consulter la [documentation officielle d'Open Web UI \(GitHub\)](#) ([🏠 Home | Open WebUI](#)).



## 5 Fine-Tuning des Modèles

### 5.1 Recherche et Exploration du Fine-Tuning

Pour améliorer la performance de nos modèles de langage, nous avons exploré diverses méthodes de fine-tuning et découvert la bibliothèque Unsloth. Unsloth est une bibliothèque légère qui permet de réaliser le fine-tuning des LLM (Large Language Models) de manière plus rapide et plus efficace en termes de mémoire. Elle est compatible avec l'écosystème Hugging Face, incluant les bibliothèques transformers, PEFT, et TRL.

#### Caractéristiques Principales d'Unsloth :

- **Optimisation de la Mémoire** : Unsloth réduit l'utilisation de la mémoire jusqu'à 80%, ce qui permet d'effectuer le fine-tuning sur des machines avec des ressources limitées.
- **Vitesse Accrue** : Elle accélère le processus de fine-tuning de 2 à 5 fois par rapport aux méthodes traditionnelles, sans perte de précision.
- **Compatibilité** : La bibliothèque prend en charge les GPU NVIDIA depuis 2018 et fonctionne sur les systèmes Linux et Windows via WSL (Windows Subsystem for Linux).

### 5.2 Utilisation de Google Colab pour le Fine-Tuning sur un Dataset en Anglais (Alpaca)

Nous avons utilisé Google Colab pour effectuer le fine-tuning en raison de ses capacités de traitement GPU gratuites et de son intégration facile avec Unsloth. Voici les étapes que nous avons suivies :

#### 1. Préparation de l'Environnement :

- Création d'un environnement conda spécifique pour Unsloth.
- Installation des dépendances nécessaires, y compris PyTorch, cudatoolkit, xformers, et Unsloth lui-même.

## 2. Chargement des Modèles et des Données :

- Utilisation de la fonction `FastLanguageModel.from_pretrained` pour charger le modèle pré-entraîné depuis Hugging Face. Cette méthode permet également de charger le tokenizer et de configurer le modèle pour un fine-tuning quantifié en 4 bits, ce qui réduit encore l'utilisation de la mémoire.
- Chargement du dataset Alpaca via `datasets.load_dataset`, un ensemble de données couramment utilisé pour l'entraînement et le fine-tuning de modèles de langage.

## 3. Fine-Tuning avec QLoRA (Quantized Low-Rank Adaptation) :

- QLoRA : C'est une technique d'adaptation de bas-rang quantifié qui réduit la complexité des calculs en utilisant des approximations de rang bas, optimisant ainsi la mémoire utilisée pendant le fine-tuning sans dégrader la précision du modèle.
- Utilisation de la méthode `get_peft_model` d'Unsloth pour ajouter des adaptateurs LoRA au modèle chargé. LoRA (Low-Rank Adaptation) permet de modifier uniquement une petite fraction des paramètres du modèle, ce qui rend le fine-tuning plus efficace et moins gourmand en ressources.
- Entraînement du modèle en utilisant `SFTTrainer` de la bibliothèque TRL, qui permet un fine-tuning supervisé efficace en combinant des

méthodes de gradient accumulation et de checkpointing pour une meilleure gestion de la mémoire.

## 5.3 Défis Techniques et Résolutions

### 1. Problèmes d'Installation sous Windows et Solution via WSL :

- **WSL (Windows Subsystem for Linux)** : Permet d'exécuter un environnement Linux sur un système Windows, résolvant ainsi les problèmes de compatibilité et d'installation des bibliothèques nécessaires. WSL est crucial pour développer et exécuter des applications nécessitant un environnement Linux tout en utilisant un système Windows.
- **Installation de WSL** : Activation de WSL sur Windows et installation d'une distribution Linux (Ubuntu), suivie de la configuration de l'environnement Linux pour le développement Python.

### 2. Adaptation du Code de Fine-Tuning pour Utiliser un Dataset Local :

- Modification du code pour charger les fichiers de dataset locaux en utilisant `load_dataset` avec un chemin de fichier local.
- Adaptation des chemins dans les configurations de l'application pour pointer vers les fichiers locaux.
- Exécution de tests pour vérifier que le modèle pouvait correctement charger et traiter les données locales, et validation des résultats pour s'assurer que les performances du modèle étaient conformes aux attentes.

Ces étapes ont permis de surmonter les défis techniques et d'optimiser notre processus de fine-tuning, rendant notre approche plus robuste et efficace.

## 6 Problèmes et Solutions lors du Fine-Tuning

### 6.1 Entraînement Spécifique et Création de Dataset

Pour illustrer le processus de fine-tuning, nous avons entrepris un projet visant à entraîner un modèle de langage à parler comme un Schtroumpf (Voir [Annexe IV](#)). Cette tâche nous a permis d'explorer les techniques de création et de modification de datasets et d'adapter le style de réponse du modèle.

#### 1. Création du Dataset avec ChatGPT 4 :

Nous avons utilisé ChatGPT 4 pour générer des dialogues et des phrases dans le style des Schtroumpfs. Les résultats ont été rassemblés pour former un dataset initial.

#### 2. Modification Manuelle du Dataset :

Après avoir obtenu le dataset de base, nous avons effectué des modifications manuelles pour améliorer la cohérence et la qualité des données. Cela incluait la correction des erreurs, l'ajout de variations linguistiques et l'alignement des dialogues sur le style attendu.

### 6.2 Utilisation et Transformation des LoRAs

L'utilisation des adaptateurs LoRA (Low-Rank Adaptation) a présentée des défis techniques, notamment la gestion des fichiers de sortie après le fine-tuning.

- **Problème** : Le dossier de sortie contenant le modèle LoRA ne pouvait pas être utilisé directement avec Ollama.
- **Solution** : Nous avons utilisé "llama.cpp" pour transformer ces fichiers en un format plus pratique, comme gguf.

### 6.2.1 Transformation du LoRA en gguf à l'aide de "llama.cpp"

- **Processus de Transformation :**
  - **Installation et Configuration :** Nous avons installé "llama.cpp" et utilisé des scripts spécifiques pour convertir les fichiers LoRA en format gguf.
  - **Vérification :** Nous avons vérifié l'intégrité et la performance des modèles convertis pour garantir leur utilisation ultérieure.

### 6.2.2 Création et Utilisation des Modelfiles Spécifiques à Ollama

- **Création des Modelfiles :** Nous avons développé des scripts pour générer des Modelfiles intégrant les adaptateurs LoRA, configurant les paramètres pour une compatibilité optimale avec Ollama.
- **Utilisation des Modelfiles :** Ces fichiers ont été chargés dans Ollama pour des tests et des validations, avec des ajustements basés sur les retours d'expérience.

## 6.3 Problèmes Résolus et Nouveaux Défis

Tout au long du projet, nous avons documenté les solutions apportées à chaque problème rencontré.

- Guide de Transformation LoRa vers gguf :
  - Instructions détaillées sur l'utilisation de "llama.cpp" pour la conversion des modèles.
  - Exemples de scripts et de configurations utilisés.
- Création de Modelfiles :

- Processus de création et de déploiement des Modelfiles spécifiques à Ollama.
- Meilleures pratiques pour l'intégration des modèles dans Ollama.

## 6.4 Explications des Concepts Techniques

### 1. LoRA (Low-Rank Adaptation) :

**Explication** : LoRA est une technique qui injecte des matrices de décomposition de rang faible dans chaque couche d'un modèle pré-entraîné, réduisant ainsi le nombre de paramètres à entraîner et les exigences en mémoire GPU. Cela permet de rendre le fine-tuning plus efficace et moins gourmand en ressources (source: [Microsoft GitHub](#), [arXiv](#)).

**Utilité** : LoRA peut réduire le nombre de paramètres à entraîner par un facteur de 10 000 et les besoins en mémoire GPU par un facteur de 3 sans perte de performance.

### 2. gguf :

**Explication** : gguf est un format optimisé pour stocker les modèles après fine-tuning, assurant la compatibilité et facilitant le déploiement dans divers environnements comme Ollama.

### 3. "llama.cpp" :

**Explication** : Un outil qui permet de transformer les modèles LoRA en formats plus utilisables comme gguf, résolvant les problèmes de compatibilité des modèles avec certains environnements.

## 7 Extension Chrome Modifiée : Ollama-Reply

### 7.1 Contexte et Objectif

Dans le cadre de notre projet, nous avons cherché à améliorer l'interaction utilisateur avec les modèles de langage en modifiant une extension Chrome existante. À l'origine, cette extension, appelée [ollama-reply](#), permettait de répondre à des posts sur Twitter et LinkedIn en utilisant un modèle de langage. Notre objectif était de transformer cette extension pour répondre à des emails sur le site "e-partage.uha.fr", un système de gestion de courriers électroniques utilisé à l'Université de Haute-Alsace.

### 7.2 Description de l'Extension

L'extension Chrome initiale permettait de lire des tweets ou des posts LinkedIn et de générer des réponses automatiques à l'aide d'un modèle fonctionnant sur Ollama. Pour adapter cette extension aux emails d'e-partage.uha.fr, nous avons dû modifier le code source de l'extension, qui est développé en Electron, et utiliser Node.js pour produire un dossier `dist` à importer dans le navigateur.

### 7.3 Processus de Modification

#### 7.3.1 Analyse et Apprentissage

Nous avons commencé par nous renseigner sur le développement et la modification des extensions Chrome. Cela incluait la compréhension de la structure des projets Electron et l'utilisation de Node.js pour la compilation et le déploiement.

#### 7.3.2 Adaptation pour les Emails

Pour adapter l'extension aux emails, nous avons entrepris les étapes suivantes :

### 1. Scraping Dynamique des Emails :

- **Défi** : Les mails n'étaient pas présents dans le DOM au chargement de la page, mais étaient chargés dynamiquement.
- **Solution** : Après plusieurs tentatives infructueuses de scraping automatique, nous avons opté pour une fenêtre où l'utilisateur peut copier-coller le contenu de l'email. Cela permet à l'extension de générer une réponse automatique en utilisant le modèle Ollama. (Voir [Annexe V](#))

### 2. Modification du Code Source :

- Nous avons modifié les scripts de scraping pour extraire le contenu des emails au lieu des tweets ou des posts LinkedIn. Cela a impliqué des ajustements dans les fichiers `renderContent.ts` et les composants React pour cibler les éléments HTML spécifiques aux emails.
- L'interface utilisateur de l'extension a été ajustée pour inclure une fenêtre de copie-collage, facilitant l'interaction utilisateur.
- **Modification du Prompt pour l'API /chat** : Nous avons également adapté le prompt envoyé à l'API `/chat` d'Ollama pour qu'il soit pertinent dans le contexte des emails. L'API `/chat` d'Ollama permet de maintenir le contexte d'une conversation en incluant l'historique des messages précédents, contrairement à l'API `/generate` qui se concentre uniquement sur une requête unique sans contexte ([GitHub](#)) ([DEV Community](#)) ([GitHub](#)).



### 3. Intégration avec Ollama Distant :

- Initialement, chaque client pouvait se connecter à un serveur dédié pour les calculs. Cependant, cette solution n'était pas scalable.
- Nous avons implémenté un serveur proxy pour rediriger les requêtes API Ollama via le port 80 du serveur vers Ollama. Cela permettait une gestion plus centralisée et efficace des requêtes.

#### 7.3.3 Défis Techniques

Nous avons rencontré plusieurs défis techniques :

- **Scraping Dynamique** : La gestion du chargement dynamique des emails a nécessité des ajustements pour capturer le contenu une fois chargé.
- **Connexions API Distant** : Établir des connexions stables et sécurisées avec un serveur distant pour les calculs Ollama a été complexe. Nous avons finalement mis en place un serveur proxy pour gérer ces requêtes de manière centralisée.

#### 7.4 Résultats et Observations

Les modifications apportées ont amélioré l'utilisabilité de l'extension pour les utilisateurs de l'UHA :

- **Facilité d'Utilisation** : L'ajout de la fenêtre de copie-collage a simplifié l'interaction pour les utilisateurs finaux, permettant de générer des réponses automatiques aux emails de manière efficace.
- **Qualité des Réponses** : Les réponses générées par le modèle fine-tuné étaient pertinentes et contextuellement adaptées, bien que de temps en temps le destinataire était confondu avec l'expéditeur.

## 7.5 Conclusion

La modification de l'extension Chrome [ollama-reply](#) pour répondre aux emails sur e-partage.uha.fr a démontré la flexibilité et l'efficacité des modèles de langage fine-tunés avec LoRA dans des applications pratiques. Les résultats obtenus jusqu'à présent sont prometteurs et ouvrent la voie à des améliorations et optimisations futures pour une meilleure expérience utilisateur.

## 8 Documentation et Instructions

### 8.1 Création de Guides pour la Création Rapide de Modelfiles

La documentation est un élément crucial pour assurer que les utilisateurs puissent reproduire et adapter nos méthodes de fine-tuning et de création de Modelfiles. Nous avons développé des guides complets et accessibles pour aider les utilisateurs à naviguer dans le processus de création de Modelfiles.

#### 1. Structuration des Guides :

- Introduction : Explication des concepts de base des Modelfiles et de leur utilisation avec Ollama.
- Étapes de Création : Détails étape par étape sur la création des Modelfiles, y compris la configuration des paramètres et l'intégration des LoRAs.
- Exemples Pratiques : Fourniture de scripts et de configurations exemples pour illustrer chaque étape du processus.
- Résolution de Problèmes : Section dédiée aux erreurs courantes et à leurs solutions.

#### 2. Accessibilité et Utilisation :

- Les guides ont été rédigés en langage clair et concis pour être accessibles aux utilisateurs de tous niveaux de compétence.
- Chaque guide est accompagné de captures d'écran pour illustrer les étapes et faciliter la compréhension.

### 8.2 Amélioration Continue de la Compréhension des Modelfiles

Pour garantir que nos guides restent pertinents et utiles, nous avons mis en place un processus d'amélioration continue basé sur les retours d'expérience et les évolutions des outils.

Mise à Jour des Guides :

- Les guides sont régulièrement mis à jour pour inclure les dernières découvertes et pratiques optimales.
- Les mises à jour tiennent compte des évolutions des outils utilisés, comme les nouvelles versions de Ollama et des bibliothèques de fine-tuning.

### 8.3 Mise à Jour des Guides en Fonction des Évolutions des Outils et des Techniques

Les outils et techniques de fine-tuning évoluent rapidement. Pour rester à jour, nos guides doivent être régulièrement révisés et améliorés.

#### 1. Surveillance des Évolutions Technologiques :

- Suivi des mises à jour et des nouvelles versions des outils utilisés, comme Ollama et les bibliothèques de fine-tuning.
- Analyse des nouvelles techniques et des meilleures pratiques publiées dans la communauté de recherche et développement en intelligence artificielle.

#### 2. Intégration des Nouvelles Pratiques :

- Mise à jour des guides existants pour intégrer les nouvelles pratiques et optimiser les processus décrits.

- Ajout de nouvelles sections pour couvrir les outils et techniques émergents, assurant ainsi que nos utilisateurs ont accès aux informations les plus récentes et pertinentes.

Nos efforts de documentation et d'instruction visent à fournir des ressources complètes et accessibles, tout en maintenant une flexibilité pour intégrer les évolutions et les feedbacks. Cela garantit que les utilisateurs peuvent tirer le meilleur parti des techniques de fine-tuning et de création de Modelfiles pour leurs applications spécifiques.

## 9 Les LLM : Une Technologie en Évolution Constante

Les modèles de langage (LLM) représentent une technologie relativement récente et en constante évolution. Cette nature dynamique pose des défis uniques, notamment en matière de documentation et de ressources pédagogiques.

### 9.1 Défis de la Documentation et des Tutoriels

Étant une technologie émergente, les LLM souffrent souvent d'une documentation chaotique et incomplète. Les tutoriels avancés et les solutions spécifiques à des problèmes techniques sont rares et difficiles à trouver. Cela nous oblige à rechercher des informations éparpillées à travers divers forums, blogs, et dépôts GitHub pour assembler nous-mêmes des solutions cohérentes.

Un exemple concret de cette problématique est la création et l'utilisation des Modelfiles dans Ollama.

Les Modelfiles dans Ollama sont des fichiers de configuration qui servent de plan pour créer et partager des modèles. Ils permettent aux développeurs de définir les paramètres, les messages système, les templates, et d'autres composants essentiels. Par exemple, le fichier peut contenir des instructions comme `FROM` pour spécifier le modèle de base, `PARAMETER` pour personnaliser le comportement du modèle, et `TEMPLATE` pour définir le template de prompt envoyé au modèle.

Ces fichiers sont essentiels pour définir et gérer les modèles, mais la documentation actuelle se concentre principalement sur leur fonctionnement plutôt que sur leur création. Cela revient à connaître les ingrédients nécessaires pour faire un gâteau sans disposer de la recette. Nous avons souvent dû passer de longues heures à expérimenter et à bricoler pour parvenir à une solution fonctionnelle.

## 9.2 Évolution Technologique Continue

La rapidité avec laquelle la technologie des LLM évolue ajoute une couche supplémentaire de complexité. Par exemple, Ollama a récemment publié une mise à jour facilitant l'importation de modèles depuis Hugging Face, un site où les développeurs peuvent partager leurs modèles de langage. Cette mise à jour simplifie considérablement notre flux de travail en automatisant la conversion des modèles, ce qui est plus pratique que notre workflow précédent. Cependant, malgré cette avancée, la création des Modelfiles reste un défi en raison de la documentation limitée.

L'évolution rapide et constante des LLM présente à la fois des opportunités et des défis. Bien que de nouvelles fonctionnalités et mises à jour, comme celles d'Ollama, rendent la gestion des modèles plus accessible, elles soulignent également l'importance d'une documentation claire et complète. Nous devons souvent naviguer dans un paysage de ressources fragmentées, ce qui nécessite une grande adaptabilité et persévérance pour exploiter pleinement le potentiel de ces technologies révolutionnaires.

Entre le moment où nous avons commencé ce rapport et sa finalisation, Ollama a sorti une mise à jour notable. Désormais, il n'est plus nécessaire d'utiliser des Modelfiles pour les modèles les plus populaires lors de la conversion. Cela illustre parfaitement la rapidité et la constance de l'évolution de cette technologie.

## 10 Conclusion

Nous avons entrepris ce projet avec l'objectif de personnaliser des modèles de langage (LLM) disponibles en open-weights pour optimiser leurs performances pour des applications spécifiques. Notre démarche a été structurée autour de plusieurs étapes clés, chacune apportant des enseignements précieux et des défis techniques.

Dans la phase initiale, nous avons mené des recherches approfondies pour comprendre les concepts de base des LLM et les meilleures pratiques en matière de personnalisation. Nous avons ensuite installé et configuré des outils essentiels tels qu'Ollama et Open WebUI, qui nous ont permis de tester et d'interagir avec les modèles de langage de manière intuitive.

Lors de l'exploration des différents modèles, nous avons évalué plusieurs options, notamment les modèles Llama3, Phi-3, Mistral, Gemma, et Qwen. Les modèles Llama3, en particulier les versions 7b et 70b, se sont avérés être les plus performants et les plus adaptés à nos besoins, offrant un bon équilibre entre la qualité des réponses et l'efficacité des ressources.

Le développement de notre application de RAG (Retrieval-Augmented Generation) a été une étape cruciale, nous permettant de combiner la génération de texte avec la récupération d'informations. En utilisant des bases de données vectorielles comme Chroma, nous avons pu améliorer la pertinence des réponses générées par les modèles.

La mise en place des templates de prompts a standardisé nos interactions avec les modèles, augmentant ainsi la précision et la cohérence des réponses. De plus, le fine-tuning des modèles à l'aide de techniques avancées comme LoRA et l'utilisation de bibliothèques comme Unsloth ont permis d'optimiser les performances de nos modèles sur des tâches spécifiques.



Nous avons également modifié une extension Chrome pour adapter les réponses automatiques aux emails, démontrant ainsi la flexibilité et l'efficacité des modèles fine-tunés dans des applications pratiques. De plus, nous avons rendu l'application compatible avec des appels à une API distante pour les réponses automatiques, améliorant ainsi son utilité et son adaptabilité.

Enfin, nous avons mis un point d'honneur à documenter nos méthodes et à créer des guides accessibles pour assurer la reproductibilité et l'adaptation de nos techniques par d'autres utilisateurs.

En conclusion, ce projet a non seulement permis d'optimiser les performances des modèles de langage pour des applications spécifiques, mais il a également contribué à une meilleure compréhension des défis techniques liés à la personnalisation des LLM. Nous avons appris que l'adaptabilité, la recherche continue et une documentation claire sont essentielles pour tirer pleinement parti des technologies émergentes dans le domaine de l'intelligence artificielle et du traitement du langage naturel.

## Ressenti des Participants

### Mathias

*Tout d'abord, j'ai proposé ce projet à mon binôme, Alex, car c'était à mon goût le projet le plus intéressant parmi ceux qui étaient proposés. J'avais déjà eu l'occasion de travailler avec de gros modèles et des LoRA pour la génération d'images, mais jamais dans le cadre des LLM (Large Language Models), ce qui suscitait énormément mon intérêt. De plus, Alex et moi disposions du matériel nécessaire en termes de puissance de calcul, ce qui nous permettait d'envisager des réalisations intéressantes.*

*Une fois le projet commencé, c'était assez captivant car chaque jour, nous regardions deux heures de vidéos éducatives pour nous familiariser avec le fonctionnement des différents concepts tels que le RAG (Retrieval-Augmented*

Generation) et le fine-tuning. Progressivement, nous avons réussi à mener à bien chacun des aspects du projet, malgré certains défis techniques et des problèmes plus complexes à résoudre que d'autres.

En terminant le projet de RAG, je me suis rendu compte à quel point cela pourrait m'aider dans mes passions, car la plupart du temps, la documentation des langages est très dispersée et c'est compliqué de trouver la bonne page pour le bon problème. Le RAG pourrait m'aider à rendre cette tâche beaucoup plus rapide.

Je trouve que nous avons énormément appris au cours de ce projet. C'était très satisfaisant de voir notre application de RAG fonctionner et citer les documents correctement, ainsi que de constater que notre premier modèle fine-tuné avec notre dataset personnalisé pouvait parler comme un Schtroumpf. Enfin, j'ai trouvé particulièrement gratifiant de réussir à faire tourner un modèle Ollama sur un serveur distant, à effectuer des requêtes API et à afficher les résultats.

En conclusion, ce projet était extrêmement intéressant et j'ai beaucoup appris en le réalisant. Cette expérience a non seulement élargi mes compétences techniques mais aussi renforcé mon intérêt pour la personnalisation des modèles de langage.

## Alex

Dès que Mathias m'a proposé de prioriser ce projet, j'ai accepté sans hésitation après avoir examiné les autres options. Ce sujet, au cœur des tendances futures en informatique, m'a immédiatement captivé. Je suis convaincu que les LLMs vont révolutionner l'industrie informatique, créant potentiellement de nouveaux métiers spécialisés. Ce projet représentait donc une opportunité précieuse pour me familiariser avec les bases de cette technologie prometteuse.

Au fur et à mesure de notre avancée, j'ai rapidement perçu le potentiel immense des personnalisations possibles avec ces modèles, malgré leur légèreté comparée à des modèles tels que ChatGPT ou Claude. Avec le RAG, par exemple, Mathias m'a instantanément évoqué la possibilité de créer un modèle personnalisé pour Unreal

*Engine 5 en utilisant la documentation officielle (très lourde), pour faciliter son travail sur le logiciel. De même, le fine-tuning nous a permis de modifier en profondeur le comportement des modèles, et cela de manière relativement simple grâce à des outils comme Unsloth. Bien que la création de datasets adaptés puisse représenter un défi, il est indéniable que les possibilités sont quasi infinies et en constante évolution.*

*Sur le plan personnel, ce projet m'a permis d'acquérir des connaissances précieuses. J'ai approfondi ma compréhension des niveaux de personnalisation, que ce soit le fine-tuning, le RAG, ou encore le prompt engineering, que je connaissais déjà. J'ai également appris à télécharger et à faire fonctionner ces modèles sur ma propre machine, une compétence qui m'avait posé des difficultés par le passé, notamment lors de la sortie de Mistral. Désormais, grâce à des ressources comme Ollama et Open WebUI, je peux déployer et personnaliser ces modèles rapidement et efficacement, une compétence qui me sera certainement utile dans mon quotidien de développeur.*

*Travailler avec Mathias, passionné par ce domaine, a été particulièrement enrichissant. Nos échanges m'ont non seulement permis de me perfectionner techniquement, mais ont également nourri ma propre passion pour ces technologies. Je suis conscient que ce projet n'est qu'une étape et qu'il est crucial de continuer à me tenir informé des évolutions dans ce domaine pour rester compétitif. Je suis également convaincu que l'importance de ces technologies justifie leur intégration dans les cursus universitaires en informatique. Ce projet a été une expérience formatrice et motivante, et je suis impatient de continuer à explorer les possibilités offertes par les LLMs en général.*

## Table des illustrations

Illustration 1 : Template de prompt avec LangChain.....	21
Illustration 2 : Utilisation des templates dans Open WebUI.....	22

# Annexes

## Annexe I : Prompt utilisé pour tester les modèles

### Mes prompts

← Retour

**Titre\***

Mail assistant

**Commande\***

/ mail-assistant

Seulement caractères alphanumériques et tirets sont autorisés - Activez cette commande en tapant "/mail-assistant " à l'entrée du chat.

**Contenu du prompt\***

Je veux que tu te comportes comme un assistant pour répondre à des mails. Mon objectif est que tu puisse répondre à différents types de mails de manière concise et professionnel. Pour ce faire, tu vas devoir analyser les mails qui te seront envoyés afin de comprendre la demande associé aux mails et répondre efficacement à l'aide des pistes de réponse associé au mail, tu devras développer ces pistes de réponses et les mettre en forme pour qu'elles correspondent au contexte du mail (pro, famille...). De plus tu devras identifier le destinataire du mail à envoyer (souvent l'expéditeur du mail auquel il faut répondre) afin de le saluer au début du mail et de t'adresser à la bonne personne lors de ta réponse. Je ne te donnerais donc que des mails ainsi que des pistes de réponses pour ces mails. Et il faudra que tu me réponde par un mail uniquement. Ta réponse ne devra donc que contenir un mail sans explications supplémentaires. Pour les mails que tu génères, je veux que ton style d'écriture soit toujours humain, dynamique, facilement compréhensible et que le texte ne soit pas détectable par les détecteurs de texte générés par une intelligence artificielle. Il faudra que le mail se termine avec une formule de politesse suivi de [prenom] [nom] uniquement. Il faut ensuite une signature pour clôturer définitivement ton mail. Cette signature est souvent écrite comme suit après la formule de politesse :

--

[designation] [prenom] [nom]  
[fonction1]  
[fonction2]  
[fonction3]  
[lieuTravail], [ville], [pays]

[informationsSupp]

Tel : [tel]  
[contactSupp]  
[site]

Voici un premier exemple de mail :

[mail]

Pistes de réponse :

[reponse]

## Annexe II :

### Mail & réponse utilisé pour tester les modèles

"Objet: Demande d'informations sur vos services de conseil en marketing

Bonjour,

J'espère que vous allez bien. Je suis actuellement à la recherche de services de conseil en marketing pour mon entreprise et j'ai entendu parler de votre agence à plusieurs reprises. Je suis particulièrement intéressé par vos services et j'aimerais obtenir plus d'informations à ce sujet.

Pourriez-vous me fournir des détails sur les services que vous proposez, ainsi que des exemples de projets sur lesquels vous avez travaillé dans le passé ? De plus, j'aimerais connaître vos tarifs et vos disponibilités pour une éventuelle consultation.

Je vous remercie par avance pour votre réponse et votre aide.

Cordialement,  
Sophie Dupuis  
Directrice Marketing  
Entreprise ABC"

---

Pistes de réponse :

"Merci pour votre intérêt, on offre une gamme complète de solutions digitales (stratégie, contenu, réseaux sociaux et suivi des performances). Je vous enverrai un mail concernant les projets passés, tarifs et disponibilités. N'hésiter pas à me recontacter pour toutes questions supplémentaires !"

## Annexe III :

### Résultats des différents modèles sur le prompt pour les mails

#### Llama3:8b

Objet: Réponse à votre demande d'informations sur nos services de conseil en marketing

Bonjour Sophie Dupuis,

Je tiens à vous remercier pour votre intérêt pour les services que nous proposons à Agence Marketing XYZ. Nous offrons une gamme complète de solutions digitales (stratégie, contenu, réseaux sociaux et suivi des performances). Je vous enverrai un mail concernant les projets passés, tarifs et disponibilités. N'hésitez pas à me recontacter pour toutes questions supplémentaires !

Cordialement,

John Doe

--

M. John Doe

Conseiller Marketing

Agence Marketing XYZ, Mulhouse, France

Tel : +33 07 71 61 73 47

Email : [john.doe@agencemarketingxyz.com](mailto:john.doe@agencemarketingxyz.com)

Site web : [www.agencemarketingxyz.com](http://www.agencemarketingxyz.com)

#### Phi3:Latest

Objet : Re: Demande d'informations sur vos services de conseil en marketing

Bonjour Mme Dupuis,

Nous nous tenons à votre égard et sommes ravis de recevoir votre demande. Nous proposons une gamme complète de solutions digitales dans les domaines de stratégie, contenu, réseaux sociaux et suivi des performances. Je vous enverrai un courriel plus détaillé concernant nos projets antérieurs, ainsi que nos tarifs et disponibilités pour une consultation possible. N'hésitez pas à nous recontacter si vous avez d'autres questions ou demandes.

Cordialement,

M. [Your Name]

Conseiller Marketing

Agence Marketing XYZ

Tel : +33 07 71 61 73 47

Email : [youremail@agencemarketingxyz.com](mailto:youremail@agencemarketingxyz.com)

Site web : [www.agencemarketingxyz.com](http://www.agencemarketingxyz.com)

#### Gemma:Latest

Objet: Demande d'informations sur vos services de conseil en marketing

Bonjour Sophie,

Merci pour votre intérêt pour nos services de conseil en marketing. Nous offre une gamme complète de solutions digitales, comprenant la stratégie, le contenu, les réseaux sociaux et le suivi des performances.

Je vous enverrai un mail avec plus de détails sur nos projets de clients, tarifs et disponibilités pour une consultation. N'hésitez pas à me contacter si vous avez des questions supplémentaires.

Cordialement,

John Doe

Conseiller Marketing

Agence Marketing XYZ



**Mistral:Latest** 03/06/2024 15:50

Subject: Re: Inquiry about Marketing Consulting Services

Bonjour Sophie Dupuis, Directrice Marketing d'Entreprise ABC,

Je vous remercie de votre intérêt dans nos services de conseil en marketing. Nous offrons une gamme complète de solutions digitales (stratégie, contenu, réseaux sociaux et suivi des performances). Voici un résumé de quelques projets passés qui peuvent vous intéresser :

- Projet XYZ: Amélioration des performances du site web avec une augmentation de 25% des ventes.
- Projet ABC: Stratégie marketing et création d'une campagne de réseaux sociaux pour un produit nouveau, augmentant les ventes de 30%.

Nous proposons également de vous fournir des détails sur nos tarifs et notre disponibilité pour une éventuelle consultation. Je vais vous envoyer un mail séparé concernant ces informations. N'hésitez pas à me recontacter si vous avez d'autres questions supplémentaires.

Je vous remercie de votre attention et j'ai l'espoir de travailler avec vous bientôt,  
John Doe

Conseiller Marketing

Agence Marketing XYZ, Mulhouse, France

Tel : +33 07 71 61 73 47

Email : [john.doe@agencemarketingxyz.com](mailto:john.doe@agencemarketingxyz.com)

Site web : [www.agencemarketingxyz.com](http://www.agencemarketingxyz.com)

--

M. John Doe

Conseiller Marketing

Agence Marketing XYZ, Mulhouse, France

Tel : +33 07 71 61 73 47

Email : [john.doe@agencemarketingxyz.com](mailto:john.doe@agencemarketingxyz.com)

Site web : [www.agencemarketingxyz.com](http://www.agencemarketingxyz.com)



**Qwen:4b**

Bonjour,

Je voulais savoir si votre agence propose également des services de conseil marketing pour des entreprises qui travaillent en ligne.

Si c'est le cas, je voulais savoir si vous proposez également des services d'aide à la décision et à l'optimisation des processus et des systèmes.

Si c'est le cas, je voulais savoir si vous proposez également des services d'expertise et de conseil en matière juridique et administrative.

Enfin, je voulais savoir si vous proposez également des services d'aide à la recherche et à la collecte de données.

Merci beaucoup pour votre attention. N'hésitez pas à me contacter pour toutes questions supplémentaires !"



## Annexe IV : Résultats de prompt sur notre modèle schtroumpf

M

**You**

I want to make a pie, give me a a fun recipe!



**SmurfAdapllama3:Latest**

Oh, oh, oh! Pie time is SMURF time! Here's a Smurfberry Pie recipe that'll make your taste buds do the Smurfberry Shuffle!

Ingredients:


- 2 cups smurfberries (or blueberries)
- 1 cup sugar
- 1/4 cup cornstarch
- 1/2 teaspoon salt
- 1/2 cup unsalted butter, melted
- 2 large eggs
- 1 pie crust (homemade or store-bought)

Instructions:

1. Preheat your oven to 375°F (190°C). Line a baking sheet with parchment paper.
2. In a large bowl, combine the smurfberries, sugar, cornstarch, and salt. Let it sit for about 15 minutes, until the berries start to release their juice and the mixture becomes syrupy.
3. Stir in the melted butter, eggs, and vanilla extract (optional). Mix well!
4. Roll out the pie crust and place it in a 9-inch (23cm) pie dish. Fill the pie crust with the smurfberry mixture.
5. Bake for 40-50 minutes or until the crust is golden brown and the filling bubbles up slightly. Let the pie cool before serving. YUM!

## Annexe V :

### Résultats de réponse générée par l'extension

 **ollama-reply** ☆ Star me

Welcome to ollama-reply

Paste your text below and click the button to analyze

Paste your text here...

Generate reply

Developed by [Jose Donato](#). Code open source on [GitHub](#).

Alex Schlageter.

Generate another reply

### Generated reply

Bonjour Alex,

Je confirme ma disponibilité pour la réunion mensuelle prévue le vendredi 2 juin à 14h00 dans la salle de conférence principale. Je suis d'accord avec l'ordre du jour provisoire que vous avez établi, mais je souhaiterais ajouter un point sur le suivi des nouvelles recrues pour discuter de leurs intégrations et des moyens de les aider à s'adapter au sein de l'équipe.

Je suis impatient de discuter avec les autres membres de l'équipe et de partager les progrès réalisés sur nos projets en cours. Je vous remercie d'avance pour votre organisation et je suis sûr que cette réunion sera très utile pour notre équipe.

Bien cordialement,  
Mathias Beaupère.

Developed by [Jose Donato](#). Code open source on [GitHub](#).