

Travail de Bachelor 2024

Solution d'optimisation RH pour les CMS



Source : image générée par intelligence artificielle

Etudiant : Arnaud Michelet

Professeur : Michael Schumacher

RÉSUMÉ

L'avènement des nouvelles technologies basées sur l'intelligence artificielle, avec son lot d'opportunités mais aussi de risques, nous poussent à repenser notre vision et notre organisation du travail. Dans un contexte de pénurie de personnel soignant et de charge administrative croissante, l'intelligence artificielle générative se positionne comme une solution prometteuse. Ce travail de bachelor explore l'application des modèles de langage de grande taille (LLM) pour alléger les tâches administratives des soignants, améliorant ainsi leur efficacité et les conditions de travail.

L'étude commence par une analyse approfondie des besoins et des ressources des Centres Médicaux-Sociaux valaisans. Des interviews avec le personnel soignant ont permis d'identifier les tâches administratives les plus chronophages et leur impact sur la qualité des soins. Sur cette base, nous avons conçu et développé une application utilisant un LLM pour simplifier deux tâches clés : la communication soignant-patient et le résumé des dossiers médicaux.

Le processus de développement de l'application a impliqué plusieurs phases, dont la sélection des technologies adéquates, la conception de l'architecture, le fine-tuning du LLM, et des évaluations de performance. Les résultats montrent que l'automatisation de ces tâches présente un réel potentiel de réduction du temps consacré aux tâches administratives, de diminution des erreurs et d'amélioration de la satisfaction des soignants, tout en augmentant le temps disponible pour les soins directs aux patients.

Si ce projet ne débouche que sur une preuve de concept (POC), il ouvre cependant des perspectives pour l'intégration de l'IA dans le secteur des soins de longue durée, offrant ainsi des solutions innovantes pour faire face aux défis actuels. Les recommandations formulées visent à guider de futurs développements dans cette voie, assurant une adoption sécurisée et efficace de ces technologies.

Mots-clés : IA générative, LLM, Fine-tuning, Soins longue durée, Optimisation des tâches administratives

AVANT-PROPOS

Ce projet a été réalisé dans le cadre d'un travail de bachelor de la filière informatique de gestion de la HES-SO Valais-Wallis.

Le sujet du travail émane de discussions entre Abraham Ritz, mon collègue chez IT SLD Solutions et moi-même et vise à répondre à la problématique de pénurie de personnel soignant qui guette les institutions de soins longues durées.

Ce sujet, étant directement lié à mon activité professionnelle au sein de mon entreprise et aux problématiques de nos clients, m'a beaucoup intéressé. La technologie d'intelligence artificielle générative, avec ses opportunités et ses risques, a également motivé mon choix.

L'objectif de ce travail est d'évaluer si l'utilisation de l'intelligence artificielle générative peut offrir une solution viable pour alléger les tâches administratives du personnel soignant. Ce projet vise à se familiariser avec les nouvelles technologies associées aux grands modèles de langage et à développer une application de démonstration illustrant divers cas d'utilisation. Les connaissances emmagasinées et l'application serviront de base pour de futurs développements.

REMERCIEMENTS

Je tiens à remercier toutes les personnes m'ayant soutenu durant la durée de mes études et la réalisation de ce travail de bachelor.

Je remercie plus particulièrement :

- M. Michael Schumacher, professeur à la HES-SO Valais-Wallis, pour son suivi tout au long du projet et ses conseils
- M. Gaetano Manzo, collaborateur scientifique à la HES-SO Valais-Wallis et chercheur aux *National Institutes of Health* pour son suivi, son expertise dans le domaine de l'intelligence artificielle générative et ses conseils.
- M. Abraham Ritz, responsable des systèmes d'information centraux chez IT SLD Solutions pour l'opportunité de réaliser ce travail, son soutien, ses conseils, son support technique et la relecture du rapport.
- Mes parents pour le temps consacré à la relecture de mon rapport et leurs encouragements.

Ainsi que mes camarades de classe, ma famille et mes amis pour leur soutien et leurs encouragements depuis quatre ans.

TABLE DES MATIÈRES

RÉSUMÉ	II
AVANT-PROPOS	III
REMERCIEMENTS	IV
LISTE DES TABLEAUX	X
LISTE DES FIGURES	XI
LISTE DES ABRÉVIATIONS	XIII
UTILISATION DE L'INTELLIGENCE ARTIFICIELLE COMME OUTIL	1
INTRODUCTION	2
1. MÉTHODOLOGIE	3
1.1. CONTEXTE.....	3
1.2. PROBLÉMATIQUE	3
1.3. HYPOTHÈSE	3
1.4. MÉTHODOLOGIE DE RECHERCHE.....	4
1.4.1. Phase I – État de l'art et analyses des besoins et ressources.....	4
1.4.2. Phase II – Design et développement	4
1.4.3. Phase III – Test de performance.....	4
1.4.4. Phase IV – Analyse des résultats	4
2. ÉTAT DE L'ART : UTILISATION DE LLM DANS LES SOINS	5
2.1. LARGE LANGUAGE MODEL.....	5
2.1.1. Historique	5
2.1.2. Fonctionnement.....	6
2.2. LLM ET SOINS À DOMICILE	8
2.2.1. Contexte du personnel soignant en Suisse.....	8
2.2.2. Opportunités des LLM	10
2.2.3. Limites des LLM	11
2.3. USE CASES DANS LES TÂCHES ADMINISTRATIVES DES SOINS.....	11
2.3.1. Remplissage automatique du dossier patient.....	11
2.3.2. Planification des tournées.....	12
2.3.3. Aide à la communication soignant-patient.....	12

2.3.4.	<i>Résumé du dossier patient</i>	13
2.3.5.	<i>Anonymisation du dossier patient</i>	13
2.3.6.	<i>Recherche de dossiers à l'aide de mots-clés ou images</i>	14
3.	ANALYSE DES BESOINS	14
3.1.	INTERVIEWS AUPRÈS DES COLLABORATEURS DES CMS.....	14
3.1.1.	<i>Entretien avec M. Frédéric Pralong, directeur des CMS de Sion Hérens Conthey</i> 15	
3.1.2.	<i>Entretien avec M. Arnaud Zufferey, responsable de l'aide et des soins</i>	17
3.1.3.	<i>Entretien avec M. Vincent Tamayo, infirmier</i>	19
3.2.	USE CASE SÉLECTIONNÉ	20
3.2.1.	<i>Critères de décision</i>	21
3.2.2.	<i>Diagramme de use case</i>	22
4.	ANALYSE DES RESSOURCES	23
4.1.	BASE DE DONNÉES DE L'APPLICATION DOSSIER PATIENT PRIMAIRE	23
4.2.	DOSSIER PATIENT PRIMAIRE	23
5.	CONTRAINTES DU PROJET	24
5.1.	CONFORMITÉ À LA NLPD	24
5.1.1.	<i>Protection des données</i>	24
5.1.2.	<i>Base de données médicales des CMS</i>	25
5.1.3.	<i>Entraînement du modèle</i>	25
5.1.4.	<i>Exploitation du LLM entraîné</i>	26
5.1.5.	<i>Registre de traitements</i>	26
5.2.	CONTRAINTE ARCHITECTURALE.....	27
5.3.	CONTRAINTE BUDGÉTAIRE	27
6.	CHOIX TECHNOLOGIQUES	27
6.1.	ARCHITECTURE.....	28
6.2.	LLM.....	30
6.2.1.	<i>Critères d'analyse</i>	30
6.2.2.	<i>Comparaison des LLM</i>	31
6.2.3.	<i>Choix final</i>	36
6.3.	FRAMEWORK ET LANGAGES DE PROGRAMMATION	36
6.3.1.	<i>Backend</i>	36

6.3.1.1.	<i>Django (Python)</i>	36
6.3.1.2.	<i>FastAPI (Python)</i>	37
6.3.1.3.	<i>Express.js (Node.js)</i>	37
6.3.1.4.	<i>Comparaison</i>	38
6.3.1.5.	<i>Choix final</i>	39
6.4.	FRONTEND	39
6.4.1.1.	<i>React Native</i>	39
6.4.1.2.	<i>Flutter</i>	39
6.4.1.3.	<i>Angular avec Ionic</i>	40
6.4.1.4.	<i>Comparaison</i>	41
6.4.1.5.	<i>Choix final</i>	41
6.5.	ENVIRONNEMENT DE DÉVELOPPEMENT	42
6.5.1.	<i>Visual Studio Code</i>	42
6.5.2.	<i>Pycharm</i>	42
6.5.3.	<i>IntelliJ IDEA</i>	42
6.5.4.	<i>Comparaison</i>	43
6.5.5.	<i>Choix final</i>	44
6.6.	FINE-TUNING	44
6.6.1.	<i>Stratégies de fine-tuning</i>	44
6.6.2.	<i>Full parameter Fine-tuning</i>	44
6.6.3.	<i>Adapter Tuning</i>	45
6.6.4.	<i>Low-Rank Adaptation</i>	46
6.6.5.	<i>Points d'attention</i>	47
6.6.5.1.	<i>Qualité du dataset</i>	47
6.6.5.2.	<i>Gestion des hyperparamètres</i>	47
6.6.5.3.	<i>Surapprentissage</i>	47
6.6.5.4.	<i>Besoin élevé en ressources</i>	48
6.6.6.	<i>Choix final</i>	49
6.7.	GESTION DES DONNÉES	49
7.	RISQUES DU PROJET	50
7.1.	BIAIS	50
7.2.	CYBER-ATTAQUES RÉPERTORIÉES	50
7.2.1.	<i>Attaque par empoisonnement</i>	50
7.2.2.	<i>Attaque Oracle</i>	51

7.2.3. <i>Attaque par manipulation</i>	51
7.3. HALLUCINATIONS	52
8. DÉVELOPPEMENTS	52
8.1. STRUCTURATION DES DONNÉES D'ENTRAÎNEMENT	52
8.2. CONCEPTION DES PROMPTS	53
8.2.1. <i>Prompt de vulgarisation et traduction</i>	53
8.2.2. <i>Prompt de résumé du dossier patient</i>	54
8.3. ENTRAÎNEMENT DU LLM ET INTÉGRATION.....	56
8.3.1. <i>Outils</i>	56
8.3.2. <i>Entraînement</i>	56
8.3.2.1. <i>Chargement du modèle pré-entraîné</i>	57
8.3.2.2. <i>Préparation du modèle pour un fine-tuning avec LoRA</i>	58
8.3.2.3. <i>Préparation des données d'entraînement</i>	59
8.3.2.4. <i>Définition des hyperparamètres d'entraînement</i>	60
8.3.2.5. <i>Analyse des logs d'entraînement</i>	64
8.3.3. <i>Fusion du modèle et de l'adapter</i>	65
8.3.4. <i>Conversion du modèle en .gguf</i>	67
8.3.5. <i>Création du nouveau model sur Ollama</i>	68
8.4. BACK-END : API DJANGO	69
8.4.1. <i>Structure du projet Django</i>	69
8.4.2. <i>Sécurité</i>	70
8.5. FRONT-END : FLUTTER.....	71
8.5.1. <i>Structure du projet Flutter</i>	71
8.5.2. <i>Pages de l'application</i>	72
8.5.3. <i>Multilinguisme</i>	75
8.5.4. <i>Base de données</i>	76
9. EVALUATION DU MODÈLE FINE-TUNÉ	77
9.1. EVALUATION PAR MÉTRIQUES	77
9.2. EVALUATION PAR DES PROFESSIONNELS DE LA SANTÉ.....	78
10. GESTION DE PROJET	80
11. DISCUSSION	81
12. AMÉLIORATIONS POTENTIELLES	84

12.1. VALIDATION DE LA QUALITÉ DES RÉPONSES PAR DES PROFESSIONNELS.....	84
12.2. UTILISATION DES DONNÉES DES CMS POUR LE FINE-TUNING	84
12.3. UTILISATION D'UN AUTRE SERVICE POUR L'EXPLOITATION DES LLM.....	85
12.4. CONNEXION DE L'API À L'APPLICATION DE DOSSIER PATIENT PRIMAIRE.....	85
12.5. DÉVELOPPEMENT DE NOUVELLES FONCTIONNALITÉS.....	85
CONCLUSION.....	86
ANNEXE 1 : TEMPLATE UTILISÉ POUR LES INTERVIEWS.....	91
ANNEXE 2 : REGISTRES DE TRAITEMENTS	92
ANNEXE 3 : REPOSITORY GITHUB CONTENANT LE CODE SOURCE DU PROJET .	93
ANNEXE 4 : REPOSITORY HUGGING FACE CONTENANT LES MODÈLES FINE-	
TUNÉS.....	94
ANNEXE 5 : GUIDES TECHNIQUE ET UTILISATEUR	94
ANNEXE 6 : PRODUCT BACKLOG	95
DECLARATION DE L'AUTEUR.....	96

LISTE DES TABLEAUX

Tableau 1: Configuration du bot de soutien à la rédaction	1
Tableau 2 : Sélection du use case les plus pertinents.....	21
Tableau 3 : Tableau de comparaison des LLM par les critères définis	34
Tableau 4 : Tableau de comparaison des LLM par critère et pondération	35
Tableau 5 : Tableau de comparaison des Framework de backend par critère.....	38
Tableau 6 : Tableau de comparaison des Framework de frontend par critère	41
Tableau 7: Tableau de comparaison des environnements de développement par critère	43
Tableau 8: Evaluation des métriques du modèle de base et du modèle fine-tuné	78

LISTE DES FIGURES

Figure 1: Tokenization et encodage d'un texte	6
Figure 2: Mécanisme d'attention permettant de déterminer à quel mot se réfère le "it".	7
Figure 3: Nombre d'annonces d'emploi visant du personnel soignant de niveau tertiaire. .	9
Figure 4: Nombre de diplômés dans le domaine des soins infirmiers.....	9
Figure 5: Stress perçu en % des soignants interrogés.	10
Figure 6: Vulgarisation d'un rapport médical par ChatGPT	12
Figure 7: Exemple de requête fournie à un LLM en vue d'anonymiser un rapport médical	14
Figure 8: Diagramme de use cases	23
Figure 9: Architecture simplifiée sans couche de sécurité.....	29
Figure 10 : Schéma d'infrastructure.	30
Figure 11: Ajout d'un module adapter à un modèle transformer.....	46
Figure 12: Ajustement des poids pré-entraîné avec l'ajout de matrices de rang faible	46
Figure 13: Modèle en surapprentissage.....	48
Figure 14: Modèle avec un entraînement optimal.	48
Figure 15: Besoin en RAM GPU pour charger et pour entraîner un modèle.....	49
Figure 16: Exemple d'attaque par perturbation	52
Figure 17: Echantillon représentant les 5 premières entrées du fichier json.....	53
Figure 18: Ressources à disposition sur Google Colab.....	56
Figure 19: Chargement du modèle pré-entraîné	57
Figure 20: Paramétrage de LoRA	58
Figure 21: Préparation du dataset d'entraînement	60
Figure 22: Hyperparamètres passés au SFTTrainer Source de l'auteur.....	61
Figure 23: Courbes de training et validation loss à travers les étapes.....	64
Figure 24: Fichiers composant l'adapter LoRA.....	65
Figure 25: classes python pour la fusion de l'adapter et du modèle et l'upload sur Hugging Face	66
Figure 26: Modèle final fine-tuné et disponible sur Hugging Face	67
Figure 27: Modèle fine-tuné au format gguf	67
Figure 28: LLM fine-tuné chargé sur Ollama	68
Figure 29: Structure du projet Django	69
Figure 30: Vue protégée par l'authentification par token et les permissions	71
Figure 31: Structure du projet Flutter	72

Figure 32: Page de login	73
Figure 33: Page de sign-in	74
Figure 34: Page de vulgarisation et traduction	74
Figure 35: Page de résumé du dossier patient	75
Figure 36: Fichier json contenant les paires clé-valeur en français et allemand.	76
Figure 37: Structure de la base de données	77
Figure 38: Réponse trop longue et ajout des balises "instruction" et "response"	83

LISTE DES ABRÉVIATIONS

- **API** : *Application Programming Interface* ou interface de programmation d'application
- **CHUV** : Centre hospitalier universitaire vaudois
- **CMS** : Centre Médicaux-Sociaux
- **CRUD** : *Create, Read, Update, Delete* ou créer, lire, mettre à jour, supprimer
- **DEP** : Dossier Électronique du Patient
- **EMS** : Établissements Médicaux-Sociaux
- **EPT** : Équivalent Plein Temps
- **ES** : École spécialisée
- **GPU** : *Graphical Processing Unit* ou unité de traitement graphique
- **HES** : Haute école spécialisée
- **HIPAA** : *Health Insurance Portability and Accountability Act*
- **IA** : Intelligence Artificielle
- **IT-HVS** : Département IT de l'hôpital du Valais
- **IDE** : *Integrated Development Environment* ou environnement de développement intégré
- **LLM** : *Large Language Models* ou grands modèles de langage
- **LM** : *Language Models* ou modèle de langage
- **LoRA** : *Low-Rank Adaptation*
- **MDM** : *Mobile Device Management* ou gestion des terminaux mobiles
- **MMLU** : *Massive Multitask language understanding*
- **nLPD** : Nouvelle loi sur la protection des données

- **NLP** : *Natural Language Processing* ou traitement automatique des données
- **OPAS** : Ordonnance sur les prestations de l'assurances des soins
- **ORM** : *Object-Relational Mapping* ou mapping objet-relationnel
- **POC** : *Proof Of Concept* ou preuve de concept
- **RAM** : *Random-access memory*
- **SFTTrainer** : *Supervised Fine-Tuning Trainer* ou entraîneur de fine-tuning supervisé
- **TB** : Travail de Bachelor
- **UI** : *User interface* ou Interface utilisateur
- **VM** : *Virtual Machine* ou machine virtuelle

UTILISATION DE L'INTELLIGENCE ARTIFICIELLE COMME OUTIL

Afin de réduire quelque peu le temps consacré à la rédaction du rapport et réallouer celui-ci à la partie recherche et développement du projet, l'intelligence artificielle a été utilisée comme outil de soutien rédactionnel.

Nous avons opté pour un modèle d'IA générative connu : ChatGPT dans sa version GPT-4o, à partir duquel nous avons créé un bot personnalisé nommé « Assistant pour la rédaction globale du rapport » qui a été utilisé comme aide à la rédaction pour certaines parties purement rédactionnelles.

Les prompts ont été passés sous formes de mots-clés ou courtes phrases minutieusement choisis. Toutes les réponses générées ont été scrutées avec attention et les adaptations nécessaires ont été apportées avant tout report dans le rapport.

Voici les configurations appliquées au bot :

Tableau 1: Configuration du bot de soutien à la rédaction

Nom	Assistant pour la rédaction globale du rapport
Description	Donner un sujet spécifique ainsi que des points clés ou des courtes phrases afin d'obtenir un écrit logique et structuré
Instructions	Je vais te donner des points clés relatifs à un sujet spécifique. Avec ces informations, tu devras me construire une rédaction logique et structurée, tout en respectant les normes APA. De la même manière, il t'est interdit de me fournir d'autres informations que celles explicitement fournies par les différents points clés ou courtes phrases.
Base de connaissance	Mon propre rapport au format .docx

De plus, l'IA a également été utilisé comme outil d'aide au développement logiciel, particulièrement pour le débogage. Grâce à ses capacités analytiques et à sa compréhension du code, ChatGPT a pu fournir des suggestions et des solutions pour corriger les erreurs.

INTRODUCTION

Face à une pénurie croissante de personnel soignant et à une charge administrative de plus en plus lourde dans le secteur des soins de longue durée, l'optimisation des ressources humaines est devenue une priorité. Le vieillissement de la population et l'augmentation des maladies chroniques ne font qu'exacerber ces défis, rendant urgente la recherche de solutions innovantes pour alléger le fardeau administratif pesant sur les soignants. C'est dans ce contexte que l'intelligence artificielle générative, avec ses capacités avancées de traitement du langage naturel, se présente comme une opportunité prometteuse.

Ce travail de bachelor explore la possibilité d'utiliser les modèles de langage de grande taille pour réduire la charge administrative des soignants dans les Centres Médicaux-Sociaux du Valais. En intégrant cette technologie, nous visons à faciliter certaines tâches administratives, permettant ainsi au personnel soignant de se concentrer davantage sur les soins directs aux patients.

La méthodologie adoptée pour ce projet se déploie en plusieurs phases distinctes. La première phase consiste en une analyse approfondie des besoins et des contraintes des CMS, appuyée par des interviews et des observations sur le terrain. La deuxième phase se concentre sur la conception et le développement d'une application intégrant un LLM, capable de gérer la communication soignant-patient et de résumer les dossiers patients. La troisième phase implique des tests de performance rigoureux pour évaluer l'efficacité et la sécurité de l'application. Enfin, la dernière phase se consacre à l'analyse des résultats obtenus et à la formulation de recommandations pour de futurs développements.

Ce rapport est structuré pour fournir une compréhension complète de la recherche menée et des résultats obtenus. Chaque chapitre est conçu pour détailler les étapes clés de la méthodologie, les défis rencontrés et les solutions proposées, avec pour objectif final d'apporter une contribution à l'amélioration des conditions de travail des soignants et à l'optimisation des ressources humaines dans le secteur des soins de longue durée.

1. Méthodologie

1.1. Contexte

Mon employeur, IT SLD Solutions, est une entreprise spécialisée dans la transformation numérique et la gestion des systèmes d'information dans le secteur des soins de longue durée. Elle appartient conjointement aux Centres Médicaux-Sociaux (CMS) et aux Établissements Médico-Sociaux (EMS) du canton du Valais. Ses trois axes principaux sont l'accompagnement stratégique en informatique, l'analyse commerciale, et la gestion de projet. L'entreprise gère également les systèmes d'information centraux de ces deux institutions.

De plus, IT SLD Solutions mène des projets de recherche et développement pour maintenir la veille technologique. Dans le domaine de l'intelligence artificielle, les récentes innovations soulignent des avancées notables en apprentissage profond, caractérisées par des modèles de plus en plus élaborés et des applications diversifiées, notamment dans la santé. Parallèlement, l'éthique et la protection des données personnelles restent des enjeux majeurs, poussant les professionnels à opter pour des approches responsables et transparentes.

1.2. Problématique

Le système de santé et notamment le secteur des soins longue durée se prépare à faire face, à moyen terme, à une pénurie de personnel soignant qualifié alors que la demande en prestations augmente peu à peu avec le vieillissement de la population.

De plus, une partie importante de l'emploi du temps des soignants est consacré aux tâches administratives non valorisantes et souvent perçues comme astreignantes et chronophages. Dans un contexte de croissance constante des coûts de la santé, cette charge administrative coûte cher et empêche le personnel soignant de se concentrer pleinement sur leur activité première : prodiguer des soins.

1.3. Hypothèse

L'arrivée de nouvelles technologies, notamment l'intelligence artificielle, apporte des solutions techniques permettant une réduction significative de la charge de travail des professionnels de la santé, ou encore la conversion d'équivalents temps plein (EPT) administratifs en EPT dédiés aux soins.

Sur la base de la problématique exposée, l'hypothèse peut être énoncée de la sorte : « L'utilisation d'un LLM dans le domaine des soins longue durée peut réduire considérablement la charge administrative du personnel soignant. »

1.4. Méthodologie de recherche

1.4.1. Phase I – État de l'art et analyses des besoins et ressources

Cette première phase consiste à faire un état de l'art des cas d'utilisation de LLM visant à alléger la charge administrative des métiers des soins. La pertinence des use cases identifiés dans la littérature sera ensuite jugée par le personnel soignant afin d'identifier les besoins des CMS valaisans. Un regard attentif sera aussi porté sur les ressources et les contraintes du projet afin d'entamer la phase II dans les meilleures conditions.

1.4.2. Phase II – Design et développement

Dans un premier temps, l'analyse des technologies à disposition pour réaliser les différentes étapes du projet permet définir le design de l'application. Une fois le design terminé, la phase d'entraînement du LLM et du développement de l'application débute en vue de déboucher sur un POC.

1.4.3. Phase III – Test de performance

Dans cette phase, une évaluation, basée sur plusieurs métriques, entre le LLM et après fine-tuning est réalisé pour déterminer la plus-value apportée par l'entraînement supplémentaire apporté au modèle.

1.4.4. Phase IV – Analyse des résultats

Dans cette dernière phase, les résultats sont regroupés et analysés pour évaluer de manière critique la pertinence de l'utilisation des LLM pour réduire les tâches administratives du personnel soignant. Cette phase inclut également la formulation de suggestions d'améliorations et des recommandations pour les futurs développements.

2. État de l'art : Utilisation de LLM dans les soins

2.1. Large Language Model

Les Grands Modèles de Langage ou *Large Language Models* (LLMs) sont des modèles d'intelligence artificielle spécialisés dans la compréhension et la génération de langage naturel et sont donc capables d'interagir avec le langage humain de façon cohérente et contextualisée. Ces modèles sont entraînés avec un vaste ensemble de données textuelles, ce qui leur permet de développer une compréhension profonde de la grammaire, du vocabulaire, et des relations sémantiques complexes. En plus des données purement textuelles, les LLMs sont également capables d'interpréter des données visuelles, audio et audiovisuelles (Hadi, et al., 2023).

L'intérêt majeur des LLMs réside dans leur capacité à effectuer des tâches complètement différentes comme répondre à des questions, résumer des documents ou traduire des textes dans plusieurs langues. Ils sont aussi capables de générer de vastes contenus sur la base d'instructions peu détaillées sous forme de langage humain (Amazon Web Services, Inc., 2023).

2.1.1. Historique

L'histoire des LLMs remonte aux années 1950 et les premières recherches en traitement automatique du langage (NLP). À cette époque, Shannon avait mesuré comment de simples modèles de n-grams basés sur des règles linguistiques artisanales pouvaient prédire des séquences de mots (Minaee, et al., 2024).

Dans les années 1980, les modèles linguistiques statistiques sont apparus. En utilisant des méthodes basées sur les probabilités, ceux-ci permettent d'estimer avec plus de précision des séquences de mots dans un contexte donné. Bien que plus performants que les modèles à règles, ils présentent des limites criantes dans la compréhension du contexte et de la sémantique (Hadi, et al., 2023).

C'est dans les années 2010 et l'arrivée des modèles de langage (LM) neuronaux basés sur les méthodes de *deep learning* qu'un palier majeur a été franchi. Ces modèles sont capables d'apprendre les structures du langage à partir de grands ensembles de texte (Hadi, et al., 2023).

Finalement, c'est en 2017 avec l'arrivée des modèles dits «*transformers*» qu'une compétition intense autour des LLMs s'est lancée. Ces modèles, basés sur une architecture de réseaux neuronaux avancés, se distinguent par leur mécanisme d'attention qui permet de pondérer l'importance relative de différentes parties d'une entrée de données lors du traitement du langage. Grâce à leur capacité d'entraînement parallèle sur plusieurs unités de traitement graphique (GPU), ils s'avèrent beaucoup plus rapides à entraîner que les modèles précédents et ouvrent ainsi la voie à la création de modèles beaucoup plus volumineux (Vaswani, et al., 2017).

En résumé, les LLMs et leurs impressionnants comportements résultent de plusieurs décennies de recherche sur les LM desquelles résultent 4 étapes de développements significatives : les modèles de langages statistiques, les modèles de machine learning, les modèles de deep learning et finalement les modèles transformers qui sont la base de la plupart des LLMs les plus connus.

2.1.2. Fonctionnement

Les LLM reposent sur une architecture *Transformer* qui utilise un mécanisme appelé « attention », qui permet aux modèles d'analyser le contexte d'une phrase dans son ensemble, ce qui améliore la compréhension globale du texte (Vaswani, et al., 2017).

Avant l'entraînement, les textes sont préparés pour être compréhensibles par le modèle. Cette préparation inclut la *tokenization*, où le texte est divisé en petites unités appelées *tokens*, et l'encodage, où chaque *token* est transformé en un vecteur numérique, une représentation numérique que le modèle peut comprendre et manipuler (Brown et al., 2020).

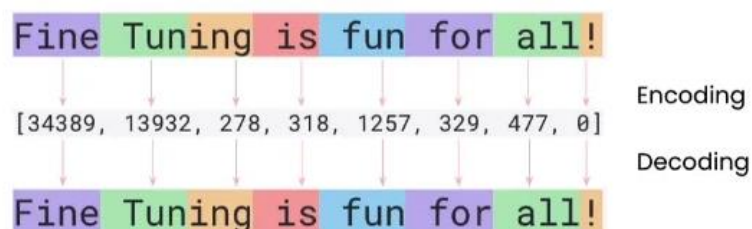


Figure 1: Tokenization et encodage d'un texte
 Source: https://miro.medium.com/v2/resize:fit:640/format:webp/1*gWP5Whykah1101EpYy17qQ.png

Le mécanisme d'attention permet quant à lui au modèle de se concentrer sur différentes parties du texte en fonction de leur pertinence. Par exemple, lorsqu'il génère un mot, le modèle peut examiner tous les mots précédents et déterminer lesquels sont les plus importants pour prédire le mot suivant (Vaswani et al., 2017). Ce processus permet au modèle de capturer des dépendances à long terme et des nuances subtiles dans le texte. Les LLM sont composés de nombreuses couches de Transformers empilées. Chaque couche traite le texte de manière légèrement différente et transmet des informations plus raffinées à la couche suivante, ce qui permet au modèle de capturer des nuances très complexes du langage (Radford, Narasimhan, & Salimans, 2018). En accumulant ces couches, le modèle peut représenter des relations de plus en plus abstraites et sophistiquées entre les tokens.

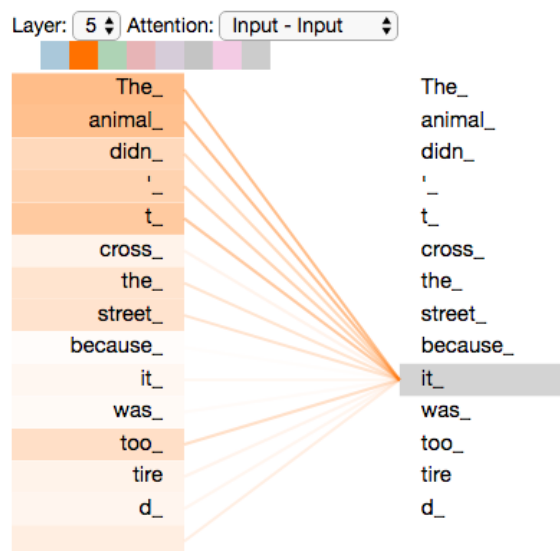


Figure 2: Mécanisme d'attention permettant de déterminer à quel mot se réfère le "it".
 Source: https://jalamar.github.io/images/t/transformer_self-attention_visualization.png

Lorsqu'on utilise le modèle pour générer du texte, on lui donne une phrase ou un début de texte, et il prédit la suite en se basant sur ce qu'il a appris. Il utilise diverses techniques pour choisir les mots les plus probables à chaque étape, tout en gardant en mémoire le contexte global de la conversation ou du texte (Brown et al., 2020). Cela permet au modèle de produire des textes cohérents et pertinents, capables de répondre de manière adéquate à une variété de demandes.

En résumé, un LLM fonctionne en divisant le texte en petites unités, en transformant ces unités en représentations numériques, puis en utilisant une série de couches pour analyser et comprendre les relations complexes entre ces unités. Grâce à l'attention et à l'entraînement

sur de vastes ensembles de données, le modèle peut générer des textes cohérents et pertinents, aidant ainsi dans de nombreuses tâches de traitement du langage naturel (Vaswani, et al., 2017); (Radford, Narasimhan, & Salimans, 2018); (Brown, et al., 2020); (Howard & Ruder, 2018).

2.2. LLM et soins à domicile

L'arrivée à maturité des LLM est perçue comme une réelle opportunité dans le domaine des soins à domicile. Outre le développement de l'aide au diagnostic qui devient de plus en plus efficace, l'IA générative représente également une grande plus-value dans l'automatisation des tâches administratives répétitives et avec peu de valeur ajoutée dans un contexte qui tend vers une pénurie de personnel soignant à moyen terme.

2.2.1. Contexte du personnel soignant en Suisse

La Suisse, comme de nombreux pays européens, fait face à une pénurie de personnel soignant exacerbée par des changements démographiques et épidémiologiques, tels que l'augmentation des maladies chroniques et des pathologies complexes due au vieillissement de la population.

Une étude réalisée par l'Observatoire suisse de la santé analyse la situation actuelle du personnel soignant dans notre pays et son évolution au cours des dernières années au travers de quelques facteurs clés (Observatoire suisse de la santé, 2024).

Le terme personnel soignant englobe 4 catégories en fonction du niveau de formation :

- **Niveau tertiaire** : infirmiers/ères diplômé(e)s.
- **Niveau secondaire II avec CFC** : assistant/e en soins et santé communautaire et assistant/e socio-éducatif.
- **Niveau secondaire II avec AFP** : aide en soins et accompagnement
- **Auxiliaires de soins** : p.ex. auxiliaire de santé de la Croix Rouge

Les résultats de cette étude démontrent que les besoins en personnel soignant sont en forte augmentation au cours de ces dernières années. En effet, les offres d'emploi dans le domaine sont passées de 44'500 en 2019 à 70'000 en 2023 (Observatoire suisse de la santé, 2024).

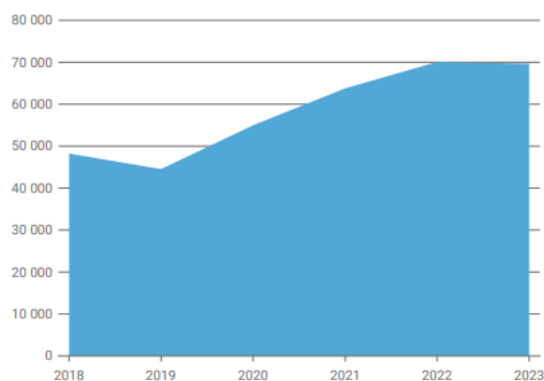


Figure 3: Nombre d'annonces d'emploi visant du personnel soignant de niveau tertiaire.
Source : (Observatoire suisse de la santé, 2024)

Dans le même temps, le nombre de diplômes délivrés par les HES et les ES dans le domaine des soins infirmiers enregistre une augmentation plus marquée et largement insuffisante pour pallier aux besoins actuels des institutions suisses de soin (Observatoire suisse de la santé, 2024).

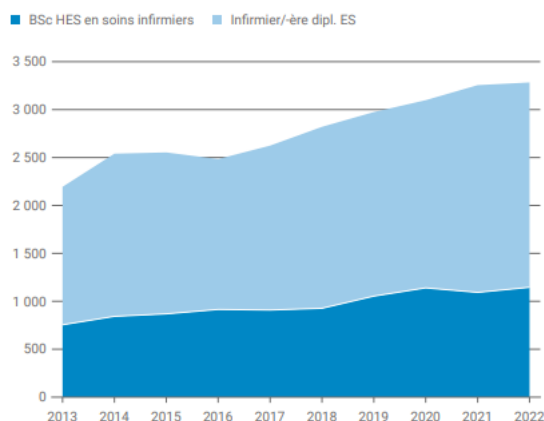


Figure 4: Nombre de diplômés dans le domaine des soins infirmiers.
Source : (Observatoire suisse de la santé, 2024)

Cette pénurie de personnel soignant entraîne des répercussions négatives sur les conditions de travail du secteur. Selon une étude de 2023 menée conjointement par Unisanté, l'Institut et Haute Ecole de la Santé La Source et le Centre hospitalier universitaire vaudois (CHUV) sur 2'287 personnes, 80% des soignants se sentent stressés, 19.2% d'entre eux travaillent dans un état de stress intense ou extrême et 13.5% se décrivent comme complètement épuisés. Ces conditions difficiles impactent négativement la qualité des soins prodigués aux patients (Observatoire suisse de la santé, 2024).

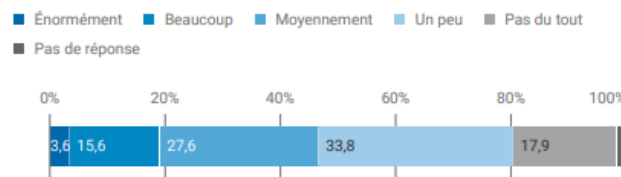


Figure 5: Stress perçu en % des soignants interrogés.
Source: (Observatoire suisse de la santé, 2024)

S'il est difficile de définir avec précision le temps alloué par les soignants aux tâches administratives, il est reconnu que la charge administrative représente une part significative du travail du personnel soignant et affecte sa disponibilité pour les soins directs aux patients. Selon l'OBSAN, trouver des moyens de décharger le personnel de santé des tâches purement administratives est un facteur clé en vue d'améliorer leurs conditions de travail et faire face à la pénurie de personnel qualifié (Observatoire suisse de la santé, 2021).

2.2.2. Opportunités des LLM

L'utilisation de LLM pour aider le personnel soignant dans les tâches administratives présente plusieurs avantages significatifs. Voici une description détaillée de ces avantages :

- **Gain de temps** : L'automatisation des tâches administratives répétitives peut permettre aux soignants de réduire drastiquement la proportion de temps passé sur les tâches sans valeur ajoutée et leur permet de consacrer davantage de temps à leurs patients (Jonas, 2023) (Nassiri & Akhloufi, 2024).
- **Réduction des erreurs** : Avec le stress et la fatigue, des erreurs de saisies de données peuvent intervenir et avoir des conséquences graves en milieu médical. L'utilisation de LLM peut permettre de réduire le risque d'erreur en automatisant la saisie et la vérification de ces données.
- **Amélioration des conditions de travail des soignants** : L'allègement de la charge administrative qui pèse sur le personnel soignant peut réduire le stress et l'épuisement. Les tâches administratives sont souvent perçues comme frustrantes et peu valorisantes par les soignants (Jonas, 2023).
- **Amélioration de la qualité des soins** : En augmentant le temps à disposition pour les soins, les soignants peuvent offrir des soins de meilleure qualité. De plus, l'utilisation de LLM peut permettre au personnel soignant d'accéder rapidement à

un vaste historique médical et leur permettre ainsi de prendre des décisions plus éclairées sur le traitement à prodiguer (Jonas, 2023) (Nassiri & Akhloufi, 2024).

- **Amélioration de la communication** : Une communication plus fluide peut avoir un effet rassurant sur le patient et lui faire accepter plus facilement les traitements proposés par le soignant (Nassiri & Akhloufi, 2024).

2.2.3. Limites des LLM

Bien que les LLM puissent automatiser de nombreuses tâches administratives, une supervision humaine reste essentielle pour garantir l'exactitude et la pertinence des informations générées. Les erreurs dans les documents médicaux peuvent avoir des conséquences graves, et les professionnels de la santé doivent vérifier et valider les informations générées par les LLM (Wang, Petzold, & Zhao, 2023).

2.3. Use cases dans les tâches administratives des soins

La littérature recense déjà plusieurs cas d'utilisation de grands modèles de langage (LLM) visant à soutenir le personnel de soins longue durée dans la réalisation de leurs tâches diverses. Ce travail se focalise exclusivement sur les cas d'utilisation de LLM comme soutien dans les tâches administratives.

2.3.1. Remplissage automatique du dossier patient

Les soignants allouent une part importante de leur temps à la retranscription dans le dossier patient primaire des informations reçues et des soins prodigués lors de leurs interventions auprès des patients. L'utilisation de LLM pour l'amélioration de l'efficacité dans les processus opérationnels est une des attentes principales des établissements de soins à domicile. Avec l'avènement des nouveaux modèles de langage entraînés pour comprendre de manière efficace un grand panel de langues et de dialectes, des nouvelles opportunités s'ouvrent pour réduire le temps dépensé par le personnel soignant sur des tâches administratives qui ont peu de valeur ajoutée. Un soignant pourrait par exemple dicter de vive voix les soins apportés à un patient à une application installée sur sa tablette afin que cette application se charge ensuite de mettre à jour de manière autonome le dossier primaire du patient et éviter ainsi un travail conséquent de retraitement (Spitex, 2024).

Un autre cas d'utilisation similaire serait l'enregistrement sur cette même application des échanges entre le soignant et le patient, afin d'identifier automatiquement les informations pertinentes et de les retranscrire automatiquement dans le dossier du patient. Cette fonctionnalité permettrait d'éviter le temps perdu lors de la phase de saisie et réduirait le risque d'erreur de saisie ou de perte d'information.

2.3.2. Planification des tournées

La planification des tournées du personnel soignant chez les patients des CMS englobe bon nombre de critères, comme le domicile du soignant et du patient, les jours effectifs de travail pour les collaborateurs à temps partiel, les vacances, la récurrence des visites et les affinités soignant-patient. Ces différents critères rendent la planification des tournées difficile pour les personnes responsables. L'utilisation d'un LLM pourrait permettre une automatisation de la planification en prenant en compte tous les critères cités ci-dessus. Cette fonctionnalité permettrait un gain de temps considérable pour les planificateurs et réduirait grandement les risques d'erreur de planning (Mfumu Kinguba & Ezzati, 2023).

2.3.3. Aide à la communication soignant-patient

La communication entre le soignant et le patient peut parfois être difficile à cause de facteurs comme la langue ou les terminologies médicales très pointues. L'utilisation d'un LLM entraîné avec des données médicales peut permettre au soignant de surmonter les barrières linguistiques et générer, dans la langue du patient, des explications synthétisées, personnalisées et vulgarisées de concepts médicaux complexes, pour différents groupes d'âge ou niveaux d'éducation. La formulation attendue de la réponse se définit dans le prompt donné au LLM (Jeblick, et al., 2023).

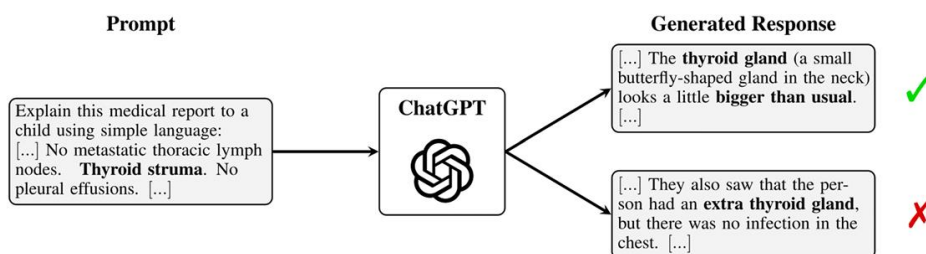


Figure 6: Vulgarisation d'un rapport médical par ChatGPT
Source : (Jeblick, et al., 2023)

2.3.4. Résumé du dossier patient

Le dossier patient est généralement volumineux et le lire dans son ensemble en vue de ressortir tous les éléments pertinents prend du temps. L'utilisation d'un LLM pourrait permettre de résumer un dossier pour ne ressortir que les informations pertinentes telles que le statut médical actuel, l'historique médical, le résultat des tests et le plan de traitement. Le résumé du dossier patient par un LLM permet non seulement un gain de temps significatif pour le soignant, mais réduit également le risque de manquer certaines informations importantes sur la condition du patient. De plus, en ayant toutes les données importantes du dossier résumé en quelques lignes, la personne qui prodigue les soins pourra identifier les meilleures actions à entreprendre. Cependant, pour que le résumé fourni par le LLM soit pertinent. (Chuang, Tang, & Hu, 2023).

2.3.5. Anonymisation du dossier patient

L'arrivée du dossier électronique du patient (DEP) a permis aux médecins et chercheurs de s'échanger facilement et de manière efficace des données médicales. L'accès à cette vaste base de données médicales est une aide précieuse pour les diagnostics ou les recherches. Cependant, cette avancée a constitué un véritable défi du point de vue de la loi sur la protection des données et l'obligation de garantir la confidentialité et l'anonymisation des données sensibles (Abu-el-rub, et al., 2022).

Dans ce contexte, les LLMs s'avèrent extrêmement efficaces pour anonymiser des rapports médicaux. Les normes HIPAA permettent de préciser, dans des prompts, quels sont les mots du rapport à anonymiser pour garantir la protection des données sensibles sur la santé des patients. (Liu, et al., 2023).

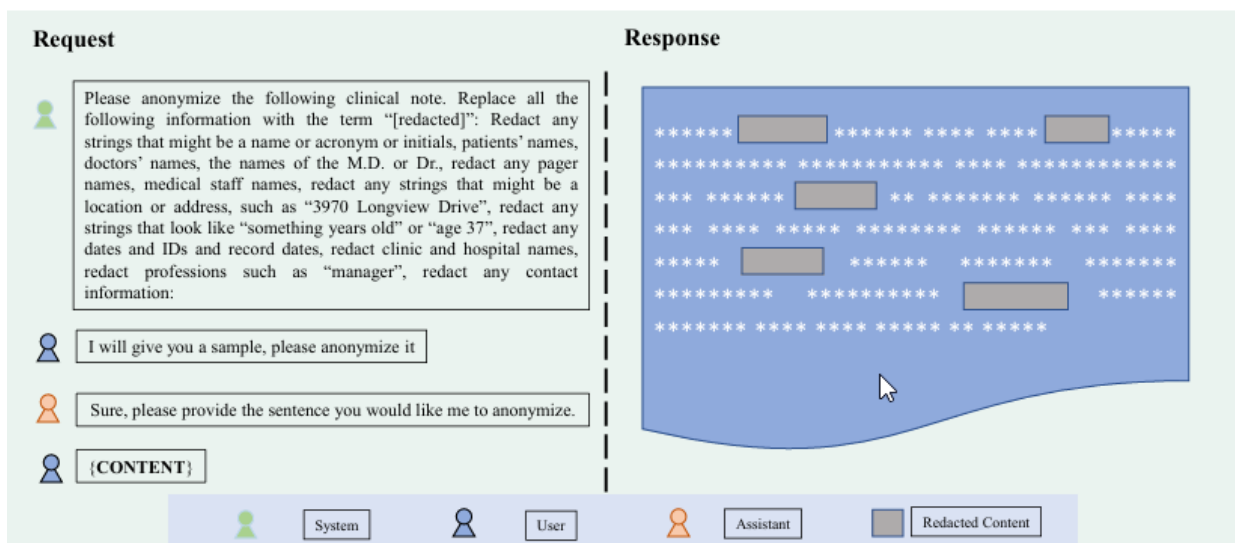


Figure 7: Exemple de requête fournie à un LLM en vue d'anonymiser un rapport médical
Source : (Liu, et al., 2023)

2.3.6. Recherche de dossiers à l'aide de mots-clés ou images

Le dossier patient primaire contient un grand nombre de cas présentant des pathologies similaires ou apparentées. Cependant, pour un soignant, il peut être difficile de retrouver un dossier comparable à celui sur lequel il travaille, en raison du grand nombre de dossiers disponibles et de leur étiquetage souvent imparfait. L'intelligence artificielle pourrait offrir une solution en permettant au professionnel de santé de localiser rapidement un cas documenté similaire, en utilisant des mots-clés ou des images. Un LLM spécifiquement entraîné à cette tâche pourrait, avec des entrées appropriées, rechercher dans toute la base de données du dossier patient primaire et extraire les dossiers les plus pertinents.

Ce cas d'utilisation nécessite une vigilance particulière quant à la gestion des droits pour éviter que des soignants accèdent à des dossiers pour lesquelles ils n'ont pas d'autorisation.

3. Analyse des besoins

Dans le but de comprendre de manière approfondie les processus métier et d'identifier les cas d'utilisation pertinents d'un LLM en vue de soulager la charge administrative du personnel soignant, nous avons rencontré plusieurs collaborateurs des CMS exerçant des métiers différents.

3.1. Interviews auprès des collaborateurs des CMS

Afin d'identifier les use cases les plus pertinents de l'utilisation d'un LLM permettant de réduire la charge administrative du personnel soignant des CMS, plusieurs interviews ont été menées auprès des métiers clés pour le projet.

Une méthodologie en 3 étapes a été appliquée à l'identique pour chaque interviewé et se présente comme suit :

1. Le contexte de mon travail de Bachelor et les objectifs de l'interview ont été décrits de manière précise aux interviewés.
2. Les use cases définis comme intéressants et qui ont été identifiés dans la littérature, ont été présentés aux interviewés pour qu'ils puissent donner leurs avis et juger de leurs pertinences dans le cadre des CMS.
3. Une discussion ouverte a été initiée pour permettre aux interviewés de partager leurs idées concernant d'éventuels use cases supplémentaires.

Le *template* utilisé pour les interviews est disponible en annexe (Annexe 1)

3.1.1. Entretien avec M. Frédéric Pralong, directeur des CMS de Sion Hérens Conthey

Date : 07.06.2024

Lieu : Bureau de M. Pralong au CMS de Sion Hérens Conthey, Chem. des Perdrix 20, 1950 Sion

Présences : Frédéric Pralong, Arnaud Michelet

Introduction de l'interviewé :

Monsieur Frédéric Pralong est le directeur des CMS de la région de Sion Hérens Conthey. Toutes les régions valaisannes utilisent le même ERP finance/comptabilité/rh et le même dossier patient primaire. Étant directeur de la plus grande région, M. Pralong possède une vision globale des processus métiers.

Contexte de l'interview :

M. Pralong a été interviewé dans les bureaux des CMS de Sion le 07.06.2024. Le but était de cibler les cas d'utilisations les plus pertinents à implémenter dans le POC qui va aboutir de ce projet. M. Pralong est directeur de la région de Sion-Hérens-Conthey depuis plusieurs

années et connaît bien les processus métier, notamment dans la partie soins, ce qui lui permet d'avoir une idée précise des besoins d'améliorations dans le domaine.

Du fait de son statut, M. Pralong est la personne habilitée à me fournir des contacts auprès du personnel soignant et la permission de les interviewer ou de les suivre dans une tournée afin d'être au plus proche du terrain et des processus métier.

Déroulement de l'interview :

Dans un premier temps, les *use cases* identifiés dans la littérature et jugé intéressants dans le cadre de ce projet ont été présentés à M. Pralong afin qu'il puisse juger de leur pertinence et faisabilité.

Ensuite, M. Pralong a eu l'opportunité de proposer des *use cases* supplémentaires qui n'auraient pas été identifiés dans la littérature mais qui seraient pertinents pour améliorer les processus métier des CMS valaisans.

Finalement, nous avons discuté de la possibilité pour moi de rencontrer des collaborateurs directement concernés par les *use cases* identifiés et qui me permettrait d'identifier plus précisément les processus métier.

Conclusion de l'interview :

Les *use cases* proposés ont tous été jugés intéressants par M. Pralong. Cependant le cas d'utilisation de remplissage automatique du dossier patient a été jugé trop risqué car des informations peuvent être manquantes ou erronées au moment de la génération automatique.

Les 3 autres *use cases* (Communication soignant-patient, résumé du dossier patient, anonymisation du dossier patient) ont été jugés pertinents et implémentables dans le cadre de mon TB.

Pour ce qui est des *use cases* supplémentaires proposés par M. Pralong, ceux-ci ne concernent pas directement le domaine des soins et sortent donc du cadre du projet. Il a cependant été discuté de les prendre en considération pour un éventuel élargissement futur du projet.

Finalement, M. Pralong m'a autorisé à interviewer des métiers directement liés aux soins (infirmier.ère, aide-soignant.e). J'ai également obtenu l'autorisation d'effectuer une tournée

avec un soignant pour identifier les processus métier. Il va lui-même sélectionner les personnes et les contacter pour les briefer.

3.1.2. Entretien avec M. Arnaud Zufferey, responsable de l'aide et des soins

Date : 25.06.2024

Lieu : Bureau de M. Zufferey au CMS de Vétroz (Coteaux du Soleil), Rle du Collège 1, 1963 Vétroz

Présences : Arnaud Zufferey, Arnaud Michelet

Introduction de l'interviewé :

M. Arnaud Zufferey est le responsable local de l'aide et des soins des Coteaux du Soleil. Il connaît parfaitement le système de soin des CMS et les différents métiers qui le composent. Au plus proche du personnel soignant, il comprend parfaitement les processus métiers et les difficultés que les soignants rencontrent.

Contexte de l'interview :

M. Zufferey a été interviewé dans les bureaux des CMS de Vétroz le 25.06.2024. Le but de l'entretien était de déterminer les cas d'utilisations les plus pertinents à implémenter pour aider le personnel soignant. Étant responsable de l'aide et des soins du CMS de Vétroz, Arnaud Zufferey connaît les difficultés rencontrées par les différents métiers des soins dans leurs tâches diverses et où se situent les améliorations potentielles.

M. Zufferey est la personne idéale pour me mettre en relation avec des soignants afin de me plonger au plus proche des processus métier.

Déroulement de l'interview :

Dans un premier temps, les *use cases* identifiés dans la littérature et jugés intéressants dans le cadre de ce projet, ont été présentés à M. Zufferey pour qu'il puisse juger de leur pertinence et faisabilité.

Ensuite, M. Zufferey a eu l'opportunité de proposer des *use cases* supplémentaires qui n'auraient pas été identifiés dans la littérature mais qui seraient pertinents pour améliorer les processus métier des CMS.

Enfin, nous avons discuté des métiers des soins pour lesquels il serait pertinent d'examiner de plus près les processus métier.

Conclusion de l'interview :

Le use case du remplissage automatique du dossier patient intéresse M. Zufferey. Actuellement, les CMS utilisent un logiciel de diction qui permet de générer du texte automatiquement à partir d'un enregistrement réalisé par un soignant. Cependant, le logiciel n'étant pas spécialisé dans le domaine médical, il ne comprend pas bon nombre de terminologies utilisées par les soignants. L'utilisation d'un LLM entraîné dans le domaine pourrait ainsi être une réelle amélioration pour les CMS.

Concernant le use case de la communication soignant-patient, un logiciel de traduction est déjà utilisé mais il comporte des limites criantes notamment chez les réfugiés parlant des langues peu courantes. De plus, celui-ci n'apporte pas de fonctionnalité de vulgarisation. L'utilisation d'un LLM peut s'avérer intéressante pour améliorer cette communication. Il convient en revanche de tester son efficacité dans les langues peu courantes sur lesquelles le modèle de base n'est pas forcément bien entraîné.

Le use case de résumé du dossier patient a également été jugé intéressant par M. Zufferey car bon nombre de dossiers sont volumineux et nécessitent beaucoup de temps pour être parcourus. Il convient en revanche d'identifier minutieusement quelles informations doivent apparaître dans le résumé et effectuer des vérifications avec le métier des soins pour s'assurer qu'aucune donnée pertinente ne soit oubliée lors de la génération du résumé par le LLM.

Le use case de l'anonymisation du dossier patient a lui aussi été jugé intéressant par M. Zufferey pour les besoins en recherche.

Arnaud Zufferey a ensuite proposé un cas d'utilisation supplémentaire. Il a souligné que les infirmières allouent d'énormes ressources à la conception et au traitement des devis de prestations envoyés aux assurances maladies. Une fois le plan de soin réalisé, un devis doit être envoyé au médecin et à l'assurance maladie. Celui-ci fait l'objet de demandes de modifications, de contestations et autres, ce qui oblige le personnel infirmier à allouer de nombreuses heures à ces tâches. M. Zufferey me fait part d'un agacement de plus en plus marqué chez le personnel infirmier, qui estime ces tâches chronophages, coûteuses en temps et sans valeur ajoutée. Pour M. Zufferey, il s'agit d'un problème qui concerne l'ensemble du

domaine médical, excepté les hôpitaux qui travaillent sur un modèle de forfait. Pour lui, un soutien de l'IA dans ce domaine serait extrêmement bénéfique et classe ce cas d'utilisation comme prioritaire.

Finalement, afin de mieux comprendre les processus métiers liés au travail infirmier, M. Zufferey me permet de passer un après-midi avec un infirmier. Cela me permettra de découvrir de manière concrètes les tâches qui occupent sa journée, notamment celles liées aux assurances maladies.

Nous avons fixé la date du 27.06.2024 de 14h à 16h dans le CMS de Vétroz (Coteaux du Soleil).

3.1.3. Entretien avec M. Vincent Tamayo, infirmier

Date : 27.06.2024

Lieu : Bureau de M. Tamayo au CMS de Vétroz (Coteaux du Soleil), Rle du Collège 1, 1963 Vétroz

Présences : Vincent Tamayo, Arnaud Michelet

Introduction de l'interviewé :

M. Vincent Tamayo est infirmier au CMS de Vétroz. Son rôle inclut la détermination des soins et services à apporter à un patient, l'administration de certains soins ainsi que la gestion des tâches administratives associées à la prise en charge d'un patient. M. Tamayo maîtrise parfaitement les processus métiers liés à la profession d'infirmier au sein des CMS.

Contexte de l'interview :

M. Tamayo a été interviewé dans les bureaux des CMS de Vétroz le 27.06.2024. Le but de l'entretien était de prendre connaissance, dans un premier temps, des processus de vérification, de modification et d'envoi des OPAS (ordonnance sur les prestations de l'assurance des soins). Puis, dans un second temps, de déterminer les cas d'utilisations les plus pertinents à implémenter pour aider le personnel soignant des CMS du Valais. Le métier d'infirmier étant au cœur du système de soin des CMS, il était intéressant, dans le cadre de ce travail, de rencontrer Vincent Tomayo.

Conclusion de l'interview :

Le traitement des OPAS s'est révélé peu pertinent dans le cadre de mon travail. Les tâches liées, allant du contrôle de l'OPAS jusqu'à la génération, en pdf, du document à envoyer au médecin et aux assurances, se font au travers de l'application Carefolio@home (dossier patient primaire utilisé par les CMS). De notre point de vue, le logiciel Carefolio@Home propose une interface et des fonctionnalités satisfaisantes. Les tâches de contrôles et de modifications des OPAS ne peuvent pas être automatisées car celles-ci nécessitent l'expertise d'un professionnel des soins. Nous avons tout de même relevé quelques *bugs* et potentielles améliorations qui seront remontées à l'éditeur de l'application pour des développements supplémentaires.

Au niveau de use cases présentés, M. Tamayo s'est montré très intéressé par le remplissage automatique du dossier patient. Selon lui, le *timing* serré entre les différentes visites chez les patients et le fait que la saisie soit effectuée sur des tablettes, contraint souvent les infirmiers ou aides-soignants à écrire rapidement des mots-clés sur les soins prodigués. Ceux-ci sont souvent vagues et parfois difficiles à comprendre quand ils sont relus ultérieurement. Pour M. Tamayo, il serait intéressant de pouvoir dicter par oral les actions qui ont été entreprises, pour qu'un LLM complète automatiquement le dossier à partir de l'enregistrement. Cette fonctionnalité permettrait de réduire le temps de saisie et d'augmenter la qualité et la compréhension des notes saisies.

Le use case de la communication soignant-patient est également pertinent pour l'interviewé. En effet, les problèmes de communications entre les soignants et les patients sont courants lorsque les discussions ont lieu avec des personnes qui ne parlent pas bien le français ou qui ne sont pas familiarisés avec les terminologies médicales.

Le use case de résumé du dossier patient est, selon M. Tamayo, intéressant mais il relève cependant que l'application Carefolio@home remonte déjà les informations les plus importantes au sommet du dossier.

Finalement, l'anonymisation du dossier patient ne concerne pas directement le travail infirmier réalisé par Vincent Tamayo.

3.2. Use Case sélectionné

La durée de réalisation du TB étant relativement courte, il a été nécessaire de choisir un use case à implémenter dans le POC du projet. Ce choix a été déterminé par les priorités données par les personnes interviewés à chaque cas d'utilisation présentés.

3.2.1. Critères de décision

Lors des entretiens, les personnes interviewées ont été invitées à classer les quatre use case présentés de 1 à 4, 1 représentant la priorité la plus élevée et 4 la plus basse. Le tableau ci-dessous regroupe les priorités données par chaque personne et présente un classement général des différents use cases.

Tableau 2 : Sélection du use case les plus pertinents

Use Case	F. Pralong	A. Zufferey	V. Tamayo	Total
Remplissage automatique du dossier patient	4	3	1	8
Communication soignant-patient	2	1	2	<u>5</u>
Résumé du dossier patient	1	2	3	<u>6</u>
Anonymisation du dossier patient	3	4	4	11

Bien que les quatre use cases présentés lors des trois entretiens réalisés aient été jugés pertinents, la fonctionnalité d'aide à la **communication patient-soignant**, ayant obtenu un total de 5 points, a été considérée comme prioritaire par les trois membres des CMS interviewés.

Ce use case a été décrit comme pertinent car le personnel soignant des CMS prend en charge un large éventail de personnes de tous âges, de cultures différentes, de langues différentes et d'horizons sociaux différents. Les collaborateurs des CMS reçus en entretien ont tous fait part de difficultés récurrentes de communication avec des patients.

Actuellement, un outil de traduction est déjà utilisé pour converser avec des patients ne parlant pas le français ou l'allemand, mais celui-ci se montre peu performant dans certaines langues, notamment l'ukrainien.

De plus, même lorsque la traduction est réalisée de manière satisfaisante, les terminologies médicales parfois spécifiques utilisées par les infirmiers ne sont pas comprises, ou le sont de manière erronée par les patients.

Ces malentendus peuvent causer des risques graves pour la qualité des soins et la sécurité des patients. Par exemple, une mauvaise compréhension des instructions médicales peut amener le patient à mal interpréter les directives concernant la prise de médicaments. Cela peut conduire à une gestion incorrecte de sa maladie. De plus, une communication inefficace peut entraîner des erreurs de diagnostic ou des plans de soins inadaptés. Enfin, ces problèmes de communication augmentent le risque de litiges. Les malentendus peuvent aboutir à des plaintes ou des poursuites judiciaires contre les professionnels de santé.

Ainsi, améliorer la communication entre les infirmiers et les patients est crucial pour minimiser ces risques et assurer une prise en charge de qualité.

Le *use case* de **résumé du dossier patient** a obtenu un total de 6 points. Les dossiers patients sont souvent assez volumineux et la rédaction de ceux-ci n'est pas toujours bien ordonnée ou rédigée de façon compréhensible.

Le temps requis pour lire l'intégralité des documents est considérable, augmentant le risque de passer à côté d'informations cruciales ou de mal interpréter celles-ci, ce qui peut poser des risques pour le patient.

L'utilisation d'un LLM, avec en entrée le dossier patient sous forme de données structurées et des instructions définies pour garantir que la réponse contienne toutes les informations importantes du dossier, pourrait offrir un gain de temps significatif pour les soignants et réduire le risque d'omission de données cruciales. Il est malgré tout essentiel que cette fonctionnalité soit rigoureusement testée par des professionnels de la santé avant toute utilisation en production.

3.2.2. Diagramme de use case

Le diagramme de use case ci-dessous clarifie les fonctionnalités de l'application et liste les acteurs avec leurs différents besoins spécifiques dans le cadre des cas d'utilisation d'aide à la communication soignant-patient et au résumé du dossier patient.

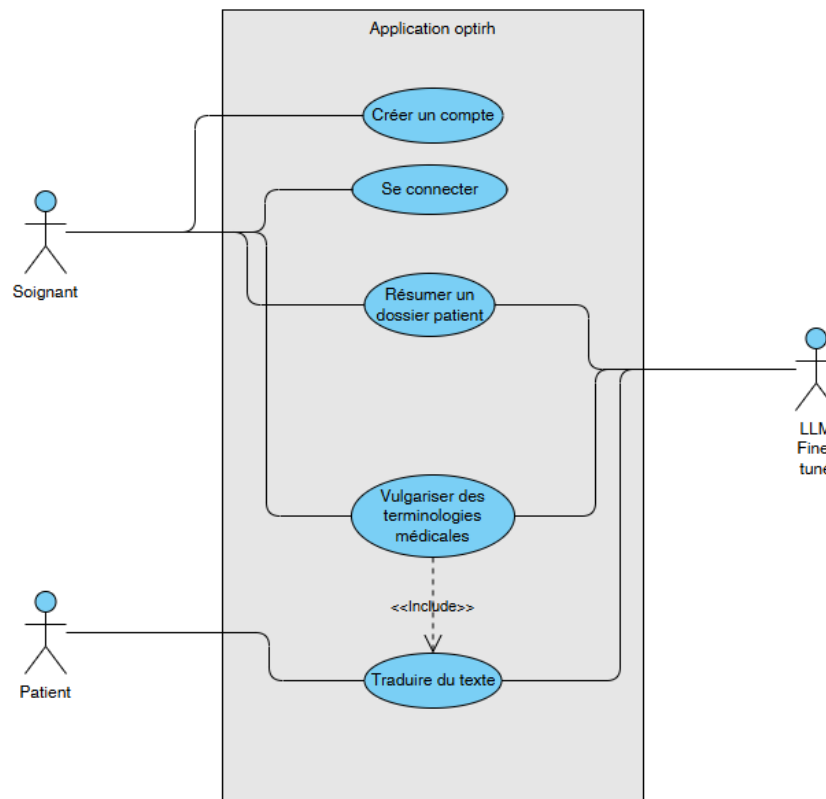


Figure 8: Diagramme de use cases
Source de l'auteur

4. Analyse des ressources

Les principales ressources à disposition pour ce projet sont les données des patients des CMS.

4.1. Base de données de l'application dossier patient primaire

Un dump de la base de données relationnelle du dossier patient nous a été mise à disposition pour les besoins de ce projet. La base de données en production est hébergée par le département informatique de l'hôpital du Valais (IT-HVS). Afin de pouvoir exécuter des requêtes SQL sur cette base, et extraire les données nécessaires, nous avons demandé le schéma relationnel de cette base au propriétaire de l'application. Malheureusement, il n'a pas été possible d'obtenir celui-ci, ce qui rend une exploitation efficace de cette ressource difficile.

4.2. Dossier patient primaire

Dans le cadre de mon travail chez IT SLD Solutions, et en respectant le secret de fonction, j'ai accès à l'application du dossier patient primaire des CMS. Cet accès me permet d'extraire des dossiers patients au format .docx ou .xlsx directement depuis l'application. Cela me donne la possibilité d'utiliser ces extraits comme ressources pour implémenter les cas d'usage du projet.

5. Contraintes du projet

Plusieurs contraintes doivent être prises en compte avant de choisir les technologies à utiliser pour la réalisation du POC.

5.1. Conformité à la nLPD

Avec l'entrée en vigueur, le 1^{er} septembre 2023, de la nouvelle loi sur la protection des données (nLPD), les activités nécessitant le traitement de données personnelles doivent être soigneusement étudiées et analysées pour garantir leur conformité du point de vue de la loi. Cette vigilance revêt une importance particulière dans ce projet en raison du caractère sensible des données de santé impliquées.

5.1.1. Protection des données

La nLPD exige le respect de 7 principes fondamentaux pour garantir la protection des données personnelles :

- **Licéité** : « *Le traitement de données personnelles doit reposer sur une base légale, sur le consentement ou sur un intérêt prépondérant public ou privé.* »
- **Finalité** : « *Les données personnelles ne doivent être traitées que dans le but indiqué lors de leur collecte, qui est prévu par la loi ou qui ressort des circonstances.* »
- **Proportionnalité** : « *Le traitement des données personnelles doit être nécessaire, adéquat et le moins intrusif possible.* »
- **Exactitude** : « *Les données personnelles sont correctes et mises à jour ; si nécessaire (par ex. inexactes ou incomplètes) sont effacées ou rectifiées.* »
- **Bonne foi** : « *Les données ne doivent en principe pas être collectées et traitées à l'insu de la personne concernée ou contre sa volonté.* »

- **Transparence de la collecte** : « *La collecte de données personnelles et sa finalité doivent être reconnaissables pour la personne concernée.* »
- **Sécurité** : « *Les données personnelles doivent être protégées contre tout traitement non autorisé, par des mesures organisationnelles et techniques appropriées* :
 - *La disponibilité*
 - *L'intégrité*
 - *La confidentialité* » (Loi fédérale sur la protection des données, 2020)

5.1.2. Base de données médicales des CMS

La base de données du dossier patient primaire des CMS est hébergée par notre partenaire, l'IT-HVS.

Afin de pouvoir effectuer des requêtes sur la base de données sans risque pour la production, notre partenaire l'IT-HVS nous a mis à disposition un dump de cette base. Les données contenues dans cette DB contenant des informations médicales sensibles, il a été nécessaire de mettre en place des mesures garantissant la sécurité de l'accès à cette base. Ce dump est hébergée sur une machine virtuelle Linux hébergée dans l'infrastructure de l'IT-HVS. Il est accessible de l'extérieur du réseau de l'IT-HVS à travers un *host jump* Citrix dont l'accès est sécurisé par une double authentification. Pour ce qui est de l'accès à la VM, seuls Abraham Ritz et moi-même possédons des comptes pour nous y connecter.

5.1.3. Entraînement du modèle

Le *fine-tuning* du modèle nécessite des ressources importantes en puissance de calcul et en électricité. Les cartes graphiques spécialisées pour un tel exercice coutent très cher et l'achat de l'une d'elle représenterait un investissement trop important pour un travail de bachelor.

Il existe cependant des plateformes cloud spécialisées qui permettent de louer de la puissance de calcul ponctuellement, en fonction des tâches et de la durée de *fine-tuning* à réaliser. Si cette solution est financièrement intéressante, elle demande une attention particulière quant au traitement des données qui seront utilisées pour l'entraînement du modèle.

Afin d'utiliser cette solution peu coûteuse sans compromettre la sécurité et la confidentialité des données patients, il a été décidé que les échantillons utilisés pour l'entraînement du LLM proviendraient de *datasets* disponibles en open-source sur des plateformes fiables. En effet, même après un processus d'anonymisation bien défini, il est impossible de garantir à 100% que les données extraites de la DB du dossier patient primaire soient complètement dépourvues d'informations personnelles.

La volonté d'utiliser les données médicales des CMS n'est malgré tout pas écartée. Le fine-tuning du modèle avec ces données propres au CMS pourrait se faire dans la continuité du projet. Il sera alors obligatoire d'acquérir de la puissance de calcul pour réaliser les étapes de fine-tuning in-house.

5.1.4. Exploitation du LLM entraîné

Lors de l'exploitation du LLM, les utilisateurs sont amenés à utiliser, dans leurs prompts, des données qui ne sont pas forcément anonymisées. Afin de répondre au mieux aux principes de *Privacy by design* et *Privacy by default*, il a été décidé d'exploiter le LLM fine-tuné sur un serveur local, directement intégré à l'infrastructure informatique des CMS, hébergée par l'IT-HVS. L'interaction avec le LLM se fait via une API sécurisée par une authentification avec token. Ce choix permet de prévenir toute fuite de données personnelles sur internet, en garantissant que les données restent sous contrôle exclusif de l'organisation. De plus, cette architecture permet de minimiser les risques d'attaques extérieures et de conserver un contrôle total sur les accès.

5.1.5. Registre de traitements

Un registre des traitements, essentiel pour la conformité à la nLPD, traite les cas d'utilisation de ce projet (Annexe 2). Ce registre est conçu pour documenter les activités de traitement de données personnelles. Au regard de la loi, celui-ci doit contenir les informations suivantes (myright.ch, s.d.):

- « *l'identité du responsable du traitement* »
- « *la finalité du traitement* »
- « *les catégories de personnes concernées et des données personnelles traitées* »
- « *les catégories de destinataires des données personnelles* »

- « dans la mesure du possible, le délai de conservation des données personnelles (ou les critères pour déterminer la durée de conservation) »
- « dans la mesure du possible, des indications concernant les mesures visant à garantir la sécurité des données »
- « en cas de communication de données personnelles à l'étranger, le nom de l'État concerné et les garanties en matière de protection des données » (Confédération Suisse, 2020)

Ce document a été élaboré à la suite d'une interview avec M. Abraham Ritz, délégué à la protection des données d'IT SLD Solutions SA et du groupement valaisan des centres médicaux-sociaux.

5.2. Contrainte architecturale

Les utilisateurs des CMS utilisent majoritairement des tablettes pour leurs tâches quotidiennes. Cette flotte de tablettes est gérée par IT SLD via une application de *Mobile Device Management* (MDM). Cette application de MDM est cruciale car elle permet de contrôler et de gérer quelles applications sont autorisées à être installées sur les tablettes, assurant ainsi un environnement sécurisé et conforme aux politiques de l'entreprise.

Il est impératif que l'application développée soit portable. Bien que les tablettes soient les principaux dispositifs utilisés, l'application doit également être pleinement fonctionnelle sur des PC. Cette portabilité assure aux utilisateurs une expérience fluide et agréable, quel que soit l'appareil utilisé.

En résumé, bien que l'accent soit mis sur les tablettes, la compatibilité de l'application sur différents types de terminaux est une exigence essentielle pour répondre aux divers besoins des collaborateurs des CMS.

5.3. Contrainte budgétaire

Les GPU disposant de la puissance de calcul nécessaire pour fine-tuner les LLM coûtent très cher (environ 5'000.- pour un Nvidia A40 par exemple). Afin de mener à bien ce travail, il va falloir réfléchir à des solutions alternatives pour réaliser le fine-tuning.

6. Choix technologiques

6.1. Architecture

Pour garantir des performances optimales, une accessibilité confortable pour les utilisateurs, et un niveau de sécurité élevé, il a été nécessaire de réfléchir à une architecture adaptée pour ce projet. Voici les composants de l'architecture :

Application frontend : Les utilisateurs s'authentifient et interagissent avec l'application via une interface utilisateur (application de frontend). Les requêtes des utilisateurs sont transmises au serveur de backend via des appels API.

Serveur API backend : Le serveur de backend reçoit les requêtes des utilisateurs, vérifie la légitimité de l'authentification. Si celle-ci est autorisée, il détermine si une interaction avec le LLM ou la base de données est demandée. Pour les requêtes nécessitant un traitement par le LLM, le backend envoie une requête au serveur LLM Ollama. Pour les opérations CRUD (Create, Read, Update, Delete) sur les données, le backend interagit avec la base de données SQL.

Serveur LLM Ollama : Le serveur LLM Ollama reçoit les requêtes du backend et utilise le LLM fine-tuné pour traiter les données et retourner une réponse. Les réponses du LLM sont renvoyées au serveur API de backend.

Base de données SQL : Le serveur API de backend interagit avec la base de données SQL pour lire et écrire des données. La base de données stocke toutes les données nécessaires au fonctionnement de l'application.

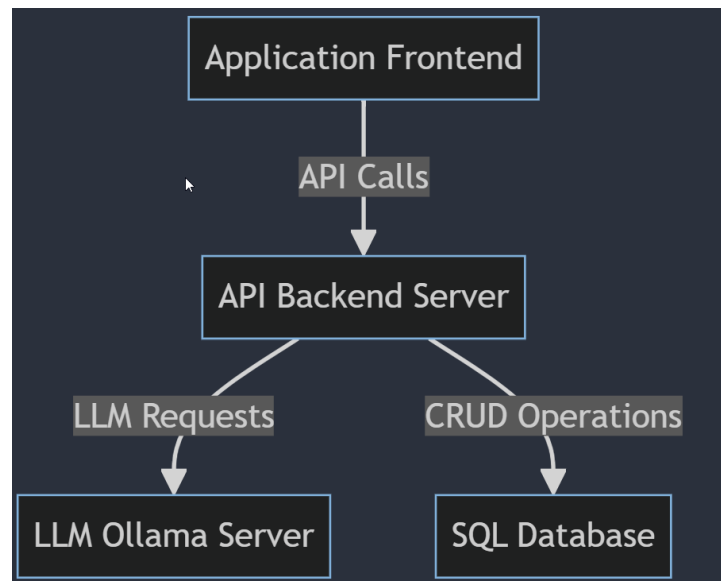


Figure 9: Architecture simplifiée sans couche de sécurité.
Source de l'auteur

Firewall : Des pare-feux protègent les connexions entre le backend et les *end devices* qui communiquent avec le frontend, assurant ainsi un niveau de sécurité élevé pour toutes les interactions.

Citrix : Une couche Citrix a été ajoutée pour garantir la sécurité de l'accès à l'application depuis l'extérieur du réseau de l'IT-HVS. L'accès à la ferme Citrix se fait au travers de comptes AD certifiés via le FrontStore. Ce principe est appliqué aux utilisateurs Desktop.

Pour les utilisateurs tablette, l'ensemble du trafic est routé à travers une Sentry positionnée dans la DMZ. La Sentry est une forme de VPN spécifiquement créée pour les périphériques mobiles qui changent régulièrement de réseau).

Infrastructure globale : L'infrastructure n'étant pas le sujet de mon TB, j'ai choisi d'appliquer les standards déjà utilisés dans le cadre du dossier patient primaire des CMS. Ces éléments étaient déjà existants et n'ont pas été mis en place spécialement pour ce POC.

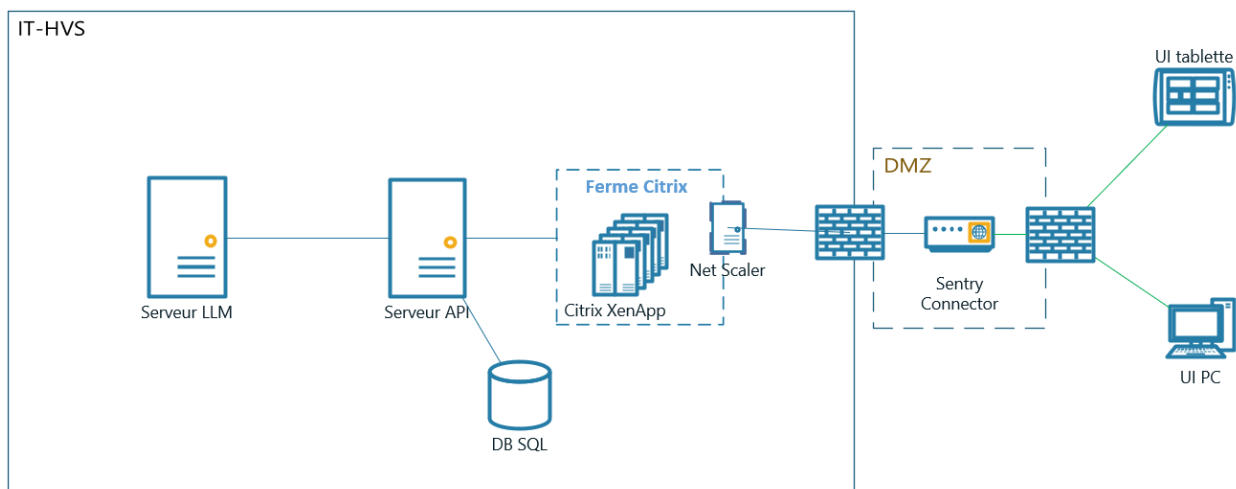


Figure 10 : Schéma d'infrastructure.
Source de l'auteur

6.2. LLM

Le choix du LLM est déterminant dans le cadre de ce travail. Le LLM est en effet le cœur de l'application développée dans le projet. De plus, compte tenu des contraintes en termes de ressources matérielles et financières, ainsi que de l'obligation de conformité à la loi sur la protection des données, faire un choix éclairé est vital pour garantir une performance acceptable avec des ressources limitées.

6.2.1. Critères d'analyse

Plusieurs critères d'analyse ont été identifiés comme pertinents en vue de choisir le LLM le plus adéquat pour le projet. Ceux-ci sont listés et décrits ci-dessous :

Open-Source : L'objectif principal de ce projet étant de fine-tuner un LLM avec les données des CMS afin de le rendre plus performant dans les tâches propres au CMS, le choix du LLM doit se porter sur un LLM open-source qui permet une exploitation, une modification et une distribution autonome et sans restriction. Ce critère étant éliminatoire, tous les LLM comparés dans la section suivante du rapport sont open-source.

Performance : La performance d'un LLM sur le benchmark MMLU (Massive Multitask Language Understanding) est un indicateur clé de sa capacité à comprendre et traiter des tâches variées et complexes. Le MMLU est le benchmark d'évaluation des LLM le plus connu et teste les modèles sur 57 tâches différentes, sur des domaines aussi variés que les

mathématiques élémentaires et le positionnement des planètes dans le système solaire. Un bon score sur ce benchmark reflète une compréhension approfondie et une capacité de résolution de problèmes dans divers contextes. Plus le score est élevé, meilleur il est.

Vitesse : La vitesse d'exécution d'un LLM est traduite par le nombre de jetons générés par secondes. Une vitesse élevée permet une meilleure expérience utilisateur, particulièrement important pour les chatbots, les assistants virtuels et les systèmes de support client. Les modèles plus petits et optimisés peuvent souvent fournir des réponses plus rapidement que les modèles très volumineux. Plus le score est élevé, meilleur il est.

Poids : Le poids du modèle est déterminé par le nombre de paramètres qu'il compte. Les modèles plus volumineux sont souvent plus performants mais plus lents et plus exigeants en termes de mémoire et puissance de calcul (RAM, GPU). Pour un modèle en 32 bits, 4GB de RAM GPU sont nécessaires pour stocker un milliard de paramètres. On se rend donc rapidement compte de l'importante capacité en calcul nécessaire pour faire stoker et entraîner un LLM.

Maîtrise du français et allemand : Le taux d'entraînement en langues française et allemande détermine dans quelle mesure le modèle maîtrise ces langues. Un modèle bien entraîné dans ces langues offrira une meilleure compréhension et génération de texte. Dans le cadre d'une utilisation pour les CMS, il est important que le modèle choisi soit suffisamment entraîné dans ces deux langues.

Prix : Le coût d'utilisation des LLM est basé sur les jetons d'entrée et de sortie. Les jetons d'entrée sont tous les jetons passés dans l'invite, tandis que les jetons de sortie sont les jetons générés. Pour certains modèles, le coût des jetons d'entrée par rapport aux jetons de sortie peut varier. Les résultats peuvent également être légèrement différents en fonction du fournisseur spécifique hébergeant le modèle.

6.2.2. Comparaison des LLM

Plusieurs LLM ont été identifiées comme pertinents dans le cadre de ce projet. Cette section présente ces LLM et les compare selon les critères définis au point précédent.

Google AI :

Google AI est la division de recherche et développement en intelligence artificielle de Google aux États-Unis. Plusieurs LLM comme Gemini 1.5 pro, Gemini 1.5 Flash et Gemma 7B ont été développés par Google. Si les deux modèles Gemini sont des modèles propriétaires, Gemma 7B est quant à lui open-source.

Gemma 7B : Il s'agit d'un modèle léger de 7 milliards de paramètres de la série Gemini. Comparé à d'autres modèles légers, il excelle dans les benchmarks académiques tels que les mathématiques, les sciences, le code, le raisonnement, le dialogue et le suivi des instructions. (Banks & Warkentin, 2024)

Meta AI :

Meta AI est la division de recherche en IA de Meta Platforms (anciennement Facebook) aux États-Unis. Meta AI a développé les modèles Llama dont les versions les plus récentes sont les modèles Llama 3. La taille de ceux-ci varie de 8 à 70 Mia de paramètres et tous sont open-source.

Llama 3 8B : Il s'agit du modèle léger de Meta AI le plus récent. Celui-ci est un peu plus volumineux que son prédécesseur Llama 2 (8 Mia de paramètres pour Llama 3 contre 7 Mia pour Llama 2). Ce modèle se révèle particulièrement performant en matière de raisonnement et compréhension de langage, de dialogue et assistance, de suivi des instructions et de génération de code. Meta AI autoproclame son modèle comme étant le modèle léger open-source le plus abouti sur le marché.

Llama 3 70B : Il est le modèle open-source le plus lourd et le plus récent de Meta AI. Comme pour Llama 8B, il se montre plus performant que la concurrence dans les tâches de raisonnement, de compréhension générale, de suivi des instructions et de génération de code. Il a été optimisé, dans son pré-entraînement, pour les cas d'utilisation en dialogue, ce qui le rend capable de générer des réponses utiles et sûres.

Mistral AI :

Mistral AI est une entreprise française spécialisée dans le développement de modèles d'intelligence artificielle de pointe. Son objectif est de développer des modèles open-source légers capables de rivaliser avec des modèles concurrents beaucoup plus lourds. Mistral AI

est également remarqué pour son engagement pour la sécurité et l'éthique tout au long des processus de développement et d'utilisation de ses modèles (Mistral AI, 2024).

Mistral 7B : Il est le modèle le plus petit de Mistral AI. Malgré sa taille, il surpasse certains modèles concurrents plus gros dans divers benchmarks. Ce modèle est particulièrement efficace dans les tâches de raisonnement, de génération de code, et de compréhension mathématique.

Mixtral 8x7B : Il s'agit du modèle de taille intermédiaire de l'entreprise française. Il est le premier modèle de Mistral à utiliser une architecture modulaire (il est composé de plusieurs sous-modèles plus petits). En effet, le modèle est composé de 8 sous-modèles de 7Mia de paramètres, chacun spécialisé dans un type de traitement. L'avantage de cette architecture est que seuls les modèles concernés par la requête vont être activés pour effectuer les tâches demandées, ce qui rend le modèle beaucoup plus léger que ses concurrents pour des performances similaires. Etant européen ce modèle maîtrise bon nombre de langues européennes, dont le français et l'allemand, bien mieux que ses concurrents américains.

Mixtral 8x22B : Modèles open-source le plus abouti de Mistral AI, Mixtral 8x22B est lui aussi basé sur une architecture modulaire avec 8 sous-modèles de 22Mia de paramètres chacun. Comme pour Mixtral 8x7B, il concurrence dans les Benchmark des modèles bien plus lourds que lui et présente une maîtrise notable du français et de l'allemand.

Maintenant que les LLM définis comme pertinents ont été listés et décrits, les tableaux ci-dessous ont pour objectif de les comparer les uns aux autres dans le but de déterminer lequel est le plus adéquat pour la réalisation du projet (MindsDB, Cerda Mardini, & Slawinska, 2024).

Tableau 3 : Tableau de comparaison des LLM par les critères définis

Modèle	Propriétaire	Performance	Vitesse	Poids en Mia de paramètres	Maîtrise fra/all	Prix (\$/1M tokens)
Gemma 7B	Google	64.3	160	7	Moyen	0.2
Llama 3 8B	Meta AI	68.4	122	8	Moyen	0.2
Llama 3 70B	Meta AI	82	49	70	Moyen	0.93
Mistral 7B	Mistral AI	62.5	74	7	Bon	0.25
Mixtral 8x7B	Mistral AI	70.6	95	12 actifs	Très bon	0.5
Mixtral 8x22B	Mistral AI	77.8	62	39 actifs	Très bon	1.2

Il convient maintenant de pondérer ces critères selon les besoins du projet afin de déterminer quel modèle choisir :

Performance : note de 1 à 3, pondération de 1

- 3 points : > 76
- 2 points : Entre 67.5 et 76
- 1 point : < 67.5

Vitesse : note de 1 à 3, pondération de 3

- 3 points : > 115.25
- 2 points : Entre 84.5 et 115.25
- 1 point : < 84.5

Poids : note de 1 à 3, pondération de 3

- 3 points : < 10
- 2 points : Entre 10 et 25
- 1 point : > 25

Prix : note de 1 à 3, pondération de 2

- 3 points : < 0.2125
- 2 points : Entre 0.2125 et 0.3750
- 1 points : > 0.3750

Maîtrise fra/all : note de 1 à 3, pondération de 2

- 3 points : Très bon
- 2 points : Bon
- 1 points : Moyen

Tableau 4 : Tableau de comparaison des LLM par critère et pondération

Modèle	Performance	Vitesse	Poids en Mia de paramètres	Maîtrise fra/all	Prix (\$/1M tokens)	Total de point avant pondération	Total de points après pondération
Pondération	1	3	1	2	2		
Gemma 7B	1	3	3	1	3	11	27
Llama 3 8B	2	3	3	1	3	12	28
Llama 3 70B	3	1	1	1	1	7	13
Mistral 7B	1	1	3	2	2	9	21
Mixtral 8x7B	2	2	2	3	1	10	22
Mixtral 8x22B	3	1	1	3	1	9	17

6.2.3. Choix final

Notre choix final se tourne vers le modèle Open-source **Llama 3 8B** de Meta AI. Disposant de ressources limitées pour l'exploitation du modèle nous forçant à nous tourner vers des modèles compacts, Llama 3 8B est le modèle le plus performant, le plus rapide et le moins couteux de sa catégorie. Il engrange, en effet, le total de points le plus élevé selon les critères et pondérations définis au chapitre précédent.

De plus, ce modèle est spécialement conçu pour du fine-tuning, ce qui le rend idéal pour ce type de projet.

6.3. Framework et langages de programmation

Cette section a pour but de comparer les différents langages de programmation et *frameworks* intéressants afin de déterminer lesquels sont les plus pertinents pour la réalisation de ce projet.

6.3.1. Backend

Plusieurs *frameworks* de backend ont des caractéristiques qui répondent aux besoins de ce travail de bachelor. Ces *frameworks* et leurs points clés sont décrits ci-après.

6.3.1.1. Django (Python)

Facilité d'utilisation : Django est reconnu pour sa productivité élevée. Il propose un cadre de travail qui permet de développer rapidement des applications web grâce à ses fonctionnalités intégrées. Parmi celles-ci, on retrouve l'ORM (*Object-Relational Mapping*), le système d'authentification robuste et le système d'administration. De plus, la maturité de ce framework lui permet d'avoir une documentation technique et des ressources communautaires vastes.

Ecosystème : L'écosystème Django est enrichi par de nombreux packages externes qui facilitent l'intégration de modèles de LLM. Des extensions comme Django REST *framework* simplifient la création d'API RESTful.

Scalabilité : Django est facilement scalable pour permettre une évolution constante de l'application.

Sécurité : La sécurité est un point crucial pour les applications manipulant des données sensibles. Django inclut plusieurs mécanismes de sécurité robustes pour protéger les données contre les menaces courantes. Il offre également une gestion de mots de passe sécurisés avec un système de hachage SHA256 et une authentification par *token*. Enfin, Django suit de près les meilleures pratiques de sécurité et publie régulièrement des mises à jour pour corriger les vulnérabilités (DataFlair, 2024).

6.3.1.2. FastAPI (Python)

Performance : FastAPI est un framework très performant. Construit sur Starlette pour les aspects web et Pydantic pour les aspects de données, ce framework est extrêmement rapide. Les benchmarks montrent que FastAPI peut rivaliser avec les frameworks les plus performants, ce qui le rend idéal pour les applications nécessitant une haute réactivité.

Modernité : FastAPI est un framework moderne et convivial pour les développeurs. Il est basé sur les annotations de type Python (*type hints*).

API : FastAPI a été spécifiquement conçu pour la création d'APIs. Il simplifie la gestion des routes, la validation des données et la documentation des APIs, tout en générant automatiquement une documentation interactive via Swagger et ReDoc. Cette spécialisation le rend particulièrement adapté pour des projets centrés sur le développement d'APIs.

Nouveauté : Fast API a été publié pour la première fois en 2018. Cette nouveauté a pour conséquences que ce framework dispose de moins de packages d'extension ou de ressources communautaires que d'autres framework plus matures (DEV, 2021).

6.3.1.3. Express.js (Node.js)

Performance : Express.js est rapide et efficace, particulièrement pour les tâches d'entrée/sortie (*I/O bound tasks*). En utilisant le moteur V8 de Google et les capacités non bloquantes de Node.js, Express.js peut gérer un grand nombre de requêtes simultanément, ce qui améliore les performances et la réactivité des applications web.

Ecosystème : Express.js bénéficie d'un large écosystème avec de nombreuses options de middleware. Cela permet aux développeurs d'ajouter facilement des fonctionnalités telles que la gestion des sessions, l'authentification, le traitement des fichiers, et plus encore. L'abondance de modules disponibles facilite l'extension des capacités de base d'Express.js pour répondre aux besoins spécifiques des projets.

JavaScript : Utiliser l'Express.js pour le backend peut rationaliser le processus de développement sur le frontend. Avoir une seule langue de programmation pour les deux côtés du développement (frontend et backend) réduit la complexité, facilite la maintenance du code, et améliore la cohérence du projet.

Callback imbriqué : Express.js peut devenir complexe à gérer avec les callbacks imbriqués, un problème connu sous le nom de "*callback hell*". Cela peut rendre le code difficile à lire et à maintenir. Cependant, ce problème peut être atténué en utilisant les Promises et les fonctionnalités *async/await* de JavaScript, qui permettent d'écrire du code asynchrone de manière plus lisible et structurée (Data Flair, 2024).

6.3.1.4. Comparaison

Le tableau ci-dessous a pour objectif de comparer les *framework* de backend identifiés comme intéressants, grâce à une liste de critères pertinents pour notre projet. Les points donnés vont de 1 à 5.

Tableau 5 : Tableau de comparaison des Framework de backend par critère

Critères	Django	FastAPI	Express.js
Ecosystème et communauté	5	3	5
Sécurité	5	4	3
Scalabilité	5	4	4
Performance	3	5	4
Rapidité de développement	4	5	4
Coûts et ressources	5	5	5
Compatibilité avec LLM	5	5	5
Familiarité personnelle avec le framework	4	2	2
Total	<u>36</u>	33	32

6.3.1.5. Choix final

Django Python est le Framework ayant recueilli le plus grand nombre de points selon les critères choisis. Les nombreux packages externes qui composent son écosystème lui procurent une excellente capacité d'intégration de LLM. De plus, Django offre une sécurité robuste avec des protections intégrées contre les attaques les plus courantes et une gestion des mots de passe efficace. Enfin, ayant déjà travaillé avec Django, mes connaissances sur ce framework me permettent de réduire considérablement le temps d'assimilation, ce qui constitue un gain de temps significatif.

6.4. Frontend

Plusieurs *frameworks* de *frontend* ont des caractéristiques qui répondent aux besoins de ce travail de bachelor. Ces frameworks et leurs points clés sont décrits ci-après.

6.4.1.1. React Native

Communauté : React Native bénéficie d'une communauté de développeurs active, ce qui permet un accès facile à des ressources, des tutoriels, et du support, facilitant ainsi la résolution de problèmes et l'optimisation des projets.

Ecosystème : Le *framework* possède un vaste écosystème de bibliothèques et de modules permettant aux développeurs d'étendre facilement les fonctionnalités de leurs applications.

Performance : les performances de React Native ne sont pas aussi élevées que celles des applications natives pures, cependant le Framework offre une expérience utilisateur fluide pour de nombreuses applications. Les performances sont suffisantes pour la majorité des cas d'utilisation.

Complexité : Bien que React Native fournisse des outils pour créer des animations, celles-ci peuvent parfois être moins fluides et plus difficiles à implémenter par rapport aux solutions natives, nécessitant des optimisations et des ajustements supplémentaires (Budny, 2024).

6.4.1.2. Flutter

Performance : Flutter offre des performances proches de celles des applications natives. Cela se traduit par une expérience utilisateur fluide et réactive, comparable à celle des applications développées directement dans les langages natifs.

UI : Grâce à son moteur de rendu graphique, Flutter permet de créer des interfaces utilisateur très personnalisées et performantes. Le *framework* offre un contrôle complet sur chaque pixel à l'écran, permettant aux développeurs de concevoir des UI attrayantes et sophistiquées, responsives sur tous types d'appareils.

Hot Reload : Cette fonctionnalité permet aux développeurs de voir instantanément les changements apportés au code sans redémarrer l'application. Hot Reload améliore considérablement la productivité en permettant un cycle de développement rapide et interactif.

Ecosystème : Bien qu'il gagne en popularité, Flutter est encore relativement jeune par rapport à des technologies plus établies comme React Native ou Angular. Cela signifie que l'écosystème des packages et des bibliothèques est moins mature, et il peut être plus difficile de trouver des solutions ou du support pour certains problèmes spécifiques (Montano, 2024).

6.4.1.3. Angular avec Ionic

Apprentissage : Pour les développeurs déjà familiers avec Angular, apprendre Ionic sera relativement simple. La similarité entre les deux *frameworks* permet une courbe d'apprentissage plus douce, facilitant l'intégration d'Ionic dans les projets existants et réduisant le temps nécessaire pour devenir productif.

Web et mobile : Ionic utilise Angular et peut cibler à la fois les applications mobiles et les applications web. Cette flexibilité permet aux développeurs de créer des applications qui fonctionnent sur plusieurs plateformes sans avoir à écrire du code spécifique pour chaque environnement.

Performance : Les applications Ionic peuvent ne pas être aussi performantes que celles créées avec des *frameworks* comme Flutter ou React Native en raison de l'utilisation de WebView. Cette dépendance à WebView peut entraîner des temps de chargement plus longs et une expérience utilisateur moins fluide.

Complexité : La gestion des dépendances et des mises à jour est plus complexe avec Angular avec Ionic. Les développeurs doivent souvent jongler avec les versions des bibliothèques et des plugins, ce qui peut ajouter une couche de complexité au développement et à la maintenance des applications (Sharma R. , 2024).

6.4.1.4. Comparaison

Le tableau ci-dessous a pour objectif de comparer les framework de frontend identifiés comme intéressants grâce à une liste de critères pertinents pour notre projet. Les points donnés vont de 1 à 5.

Tableau 6 : Tableau de comparaison des Framework de frontend par critère

Critères	React Native	Flutter	Angular / Ionic
Performance	4	5	3
Apprentissage	4	3	3
Ecosystème et communauté	5	4	4
Maintenance et scalabilité	4	4	4
Rapidité de développement	4	5	4
Compatibilité multiplateforme	4	5	3
Familiarité personnelle avec le framework	3	4	1
Total	28	<u>30</u>	22

6.4.1.5. Choix final

Flutter est le Framework ayant recueilli le plus grand nombre de points selon les critères choisis. Flutter est un *framework* puissant pour le développement d'applications multiplateformes grâce à ses performances natives, son UI riche, personnalisable et sa fonctionnalité *Hot Reload*. De plus, sa capacité à cibler le web et les applications de bureau en fait une solution très polyvalente. Finalement, ayant déjà développé avec Flutter, ma familiarité avec lui me permet de réduire considérablement le temps d'assimilation du *framework*.

6.5. Environnement de développement

Le choix de l'environnement de développement est important pour augmenter la productivité et l'efficacité du développement. Il existe plusieurs environnements de développement qui répondent aux prérequis demandés par ce projet. Voici quelques-uns d'entre eux.

6.5.1. Visual Studio Code

Visual Studio Code est un éditeur de code léger et rapide, développé par Microsoft, qui fonctionne bien même sur des machines moins puissantes. Il est hautement configurable grâce à une large gamme de thèmes, d'extensions et de paramètres de configuration. Gratuit, il prend en charge de nombreux langages de programmation et offre plusieurs extensions facilement intégrables pour Django (Django, Python) et Flutter (Flutter, Dart). L'intégration avec Git et autres systèmes de contrôle de version facilite le développement collaboratif. De plus, Visual Studio Code bénéficie d'une communauté active qui développe régulièrement des extensions et propose des ressources et des tutoriels (ClickUp, 2023).

6.5.2. Pycharm

PyCharm, développé par JetBrains, est un IDE puissant, particulièrement reconnu pour son intégration avec Django, offrant des outils avancés et une gestion efficace des projets Django. Bien que son support pour Flutter soit limité, il est possible d'y travailler via des plugins. PyCharm est plus lourd que VS Code et nécessite des machines plus puissantes pour fonctionner de manière optimale. Il propose un bon support pour les extensions et plugins, bien que ce soit plus limité comparé à VS Code. Son interface riche en fonctionnalités peut entraîner une courbe d'apprentissage plus élevée et il est moins personnalisable que VS Code. Toutefois, PyCharm bénéficie d'une grande communauté active et offre un excellent support intégré pour Git. En termes de coût, il est payant, avec une version gratuite disponible pour les étudiants (Bartra, 2024), (ClickUp, 2023).

6.5.3. IntelliJ IDEA

IntelliJ IDEA (Ultimate), lui aussi développé par JetBrains, offre un bon support pour Django, bien qu'il nécessite des configurations supplémentaires. Son support pour Flutter est excellent grâce à des extensions officielles. Performant, il est néanmoins plus lourd que VS Code. IntelliJ

dispose d'une bonne bibliothèque de plugins, et son interface utilisateur riche en fonctionnalités est similaire à celle de PyCharm. Il est hautement personnalisable, permettant aux utilisateurs d'adapter l'IDE à leurs besoins spécifiques. IntelliJ bénéficie d'une grande communauté, particulièrement active pour Java et les frameworks associés. L'intégration Git est excellente. En termes de coût, IntelliJ IDEA est payant, avec une version gratuite disponible pour les étudiants (ClickUp, 2023).

6.5.4. Comparaison

Le tableau ci-dessous a pour objectif de comparer les environnements de développement identifiés comme intéressants grâce à une liste de critères pertinents pour notre projet. Les points donnés vont de 1 à 5.

Tableau 7: Tableau de comparaison des environnements de développement par critère

Critères	Visual Studio Code	Pycharm	IntelliJ
Intégration Django	5	5	4
Intégration Flutter	5	3	5
Performance	4	3	4
Extensions/Plugins	5	4	4
Utilisation	4	4	4
Personnalisation	5	3	4
Support et communauté	5	5	4
Intégration Git	5	5	5
Prix	5	1	1
Familiarité personnelle avec l'IDE	4	1	4

Total	<u>47</u>	34	39
--------------	------------------	----	----

6.5.5. Choix final

Visual Studio Code est un choix robuste pour travailler avec Django et Flutter en raison de son excellent support des extensions, de sa performance et de sa personnalisation. Contrairement aux deux concurrents Pycharm et IntelliJ, il est totalement gratuit, en offrant des fonctionnalités équivalentes. Ayant réalisé plusieurs projets de développement avec cet outil, mon expérience sur cet environnement de développement est aussi un critère de sélection important.

6.6. Fine-tuning

Le choix d'utiliser le fine-tuning dans mon projet est justifié par les contraintes financières et matérielles, ainsi que par la nécessité de se conformer à la loi sur la protection des données.

L'objectif principal de cette approche est d'utiliser un modèle pré-entraîné léger et de l'entraîner dans un domaine spécifique, afin de parvenir à des résultats se rapprochant de ceux obtenus par des grands modèles de langage propriétaires, mais avec une empreinte beaucoup plus légère. Le fine-tuning permet d'adapter le modèle à un domaine particulier, améliorant ainsi sa pertinence et son efficacité tout en réduisant les ressources nécessaires au fonctionnement des modèles de très grande taille.

Dans le cadre ce projet, cette méthode garantit l'intégrité des données sensibles traitées en permettant l'exploitation du modèle en local sur du matériel de milieu de gamme, tout en maintenant un haut niveau de performance dans le domaine des soins.

6.6.1. Stratégies de fine-tuning

Il existe plusieurs stratégies de fine-tuning distinctes qui comportent chacune leur lot d'avantages et de désavantages.

6.6.2. Full parameter Fine-tuning

Le *full parameter fine-tuning* ou fine-tuning complet des paramètres est une stratégie qui consiste à réentraîner tous les poids du modèle de base sur un nouvel ensemble de données, dans le but de le spécialiser dans une tâche ou un domaine particulier.

Le réentraînement de tous les paramètres du modèle offre une amélioration significative de la précision et de la pertinence des résultats, en exploitant pleinement les capacités du modèle pré-entraîné. En ajustant l'ensemble des paramètres, le modèle peut mieux comprendre et traiter des informations spécifiques, assurant une adaptation fine et précise à la tâche spécifique (Lv, et al., 2024).

Cependant, cette stratégie est très coûteuse en temps et en puissance de calcul, la rendant difficilement applicable pour les structures disposant de moyens matériels limités. De plus, la gestion des hyperparamètres (paramètres d'entraînement, définis dans la section 8.3.2.4) et des ressources est complexe et nécessite une expertise avancée en fine-tuning pour éviter le surapprentissage.

6.6.3. Adapter Tuning

L'*Adapter Tuning* est une technique de fine-tuning qui permet d'adapter les LLM à des tâches ou des domaines spécifiques sans nécessiter de réentraîner tous les paramètres du modèle. Cette méthode est pratique dans des scénarios où les ressources en puissance de calcul sont limitées.

La stratégie d'*adapter tuning* consiste à entraîner seulement une petite partie du modèle de base pour créer un module supplémentaire appelé *adapter*. Cet adapter contient tous les poids mis à jour lors du fine-tuning, tandis que les poids du modèle de base restent inchangés. Une fois le fine-tuning terminé et l'adapter créé, celui-ci est fusionné avec le modèle de base pour obtenir un nouveau modèle entraîné à une tâche ou un domaine particulier.

Cette stratégie comporte plusieurs avantages. Tout d'abord, en entraînant seulement une petite partie des paramètres du modèle de base, l'*adapter tuning* est beaucoup plus rapide et bien moins coûteux en ressources de calcul que le *full parameter fine-tuning*. Ensuite, étant donné que les poids d'origine du modèle ne sont pas modifiés, les connaissances générales acquises durant le pré-entraînement sont préservées. Les *adapters* apportent simplement des ajustements nécessaires pour la tâche spécifique, sans affecter la base de connaissances du modèle. Finalement, cette méthode offre également une grande flexibilité pour adapter le modèle à diverses tâches car il est assez simple d'ajouter ou retirer des adapter à un modèle.

Cependant, étant donné que cette stratégie se focalise exclusivement sur une petite partie des paramètres, les résultats peuvent être moins performant que le *full parameter fine-tuning* (Sharma, 2023).

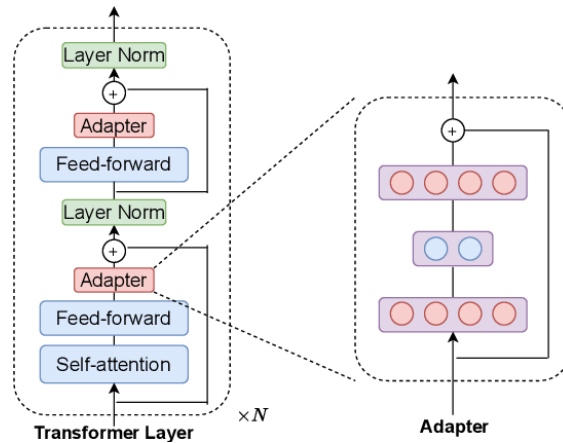


Figure 11: Ajout d'un module adapter à un modèle transformer.
Source: <https://d3i71xaburhd42.cloudfront.net/448af0627240e46df757e7b9c640ee30507c18e9/2-Figure1-1.png>

6.6.4. Low-Rank Adaptation

La stratégie Low-Rank Adaptation ou LoRA peut être considérée comme une forme d'adapter tuning car elle vise à adapter un LLM en ajoutant des modules spécialisés, entraînés à des tâches ou des domaines spécifiques.

La différence réside dans la technique utilisée. LoRA consiste à geler les poids du modèle pré-entraîné et à injecter des matrices de décomposition de rang faible (A et B) dans chaque couche de l'architecture *transformer*.

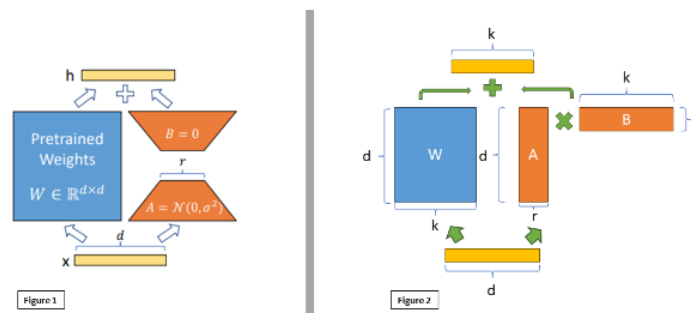


Figure 12: Ajustement des poids pré-entraîné avec l'ajout de matrices de rang faible
Source: (Hu, et al., 2021)

Cette approche réduit significativement le nombre de paramètres à ajuster et la mémoire GPU requise tout en gardant une performance proche du full parameter fine-tuning. Comme pour l'adapter-tuning, LoRA est très flexible et modulaire car il suffit de remplacer les matrices de rang faible pour changer la tâche, sans avoir besoin de réentraîner le modèle de base dans son ensemble (Hu, et al., 2021).

6.6.5. Points d'attention

Même si plusieurs outils simplifient sa mise en place, le fine-tuning reste un procédé complexe qui nécessite la prise en compte de plusieurs paramètres importants.

6.6.5.1. Qualité du dataset

La qualité des données d'entraînement est cruciale car le modèle apprend directement des exemples fournis. Des données de mauvaise qualité peuvent introduire des biais, des erreurs, et des informations non représentatives et affecter négativement les performances du modèle. Il est donc vital de vérifier la pertinence des sources d'où proviennent les données et de structurer celle-ci de manière à optimiser le fine-tuning.

6.6.5.2. Gestion des hyperparamètres

Les hyperparamètres déterminent le comportement du modèle durant l'entraînement. Une mauvaise configuration peut entraîner une performance sous-optimale ou un surapprentissage. De plus, la configuration des hyperparamètres influence grandement le besoin en puissance de calcul nécessaire pour l'entraînement du modèle. Il est donc important de comprendre l'incidence de la configuration de ces paramètres lorsque nous disposons d'une puissance de calcul limitée.

6.6.5.3. Surapprentissage

Le surapprentissage ou surentraînement intervient quand le modèle apprend trop bien les détails et le bruit des datasets d'apprentissage, perdant ainsi sa capacité à généraliser ses connaissances à de nouvelles données non vues. Un modèle victime de surapprentissage affiche généralement une haute précision sur les données d'entraînement mais peine à fournir des réponses satisfaisantes pour des données qui sortent du cadre de son dataset d'entraînement. La courbe des pertes d'entraînement et de validation est un bon moyen de

visualiser un surapprentissage. Le modèle passe généralement en surapprentissage quand le *training loss* continue de diminuer alors que le *validation loss* commence à augmenter.

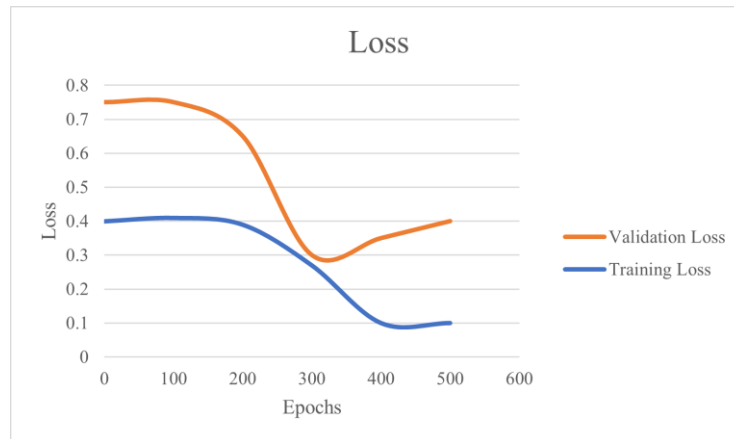


Figure 13: Modèle en surapprentissage.

Source https://www.baeldung.com/wp-content/uploads/sites/4/2022/02/img_6210aec599a86.svg

En revanche, quand les courbes de pertes convergent et se stabilisent, cela signifie que l'entraînement est optimal.

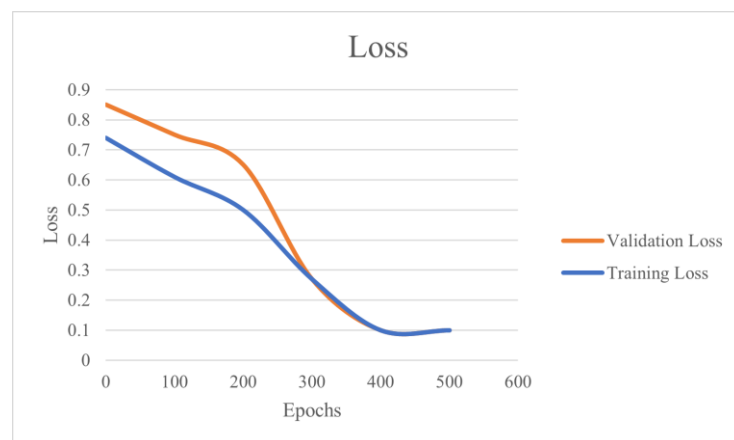


Figure 14: Modèle avec un entraînement optimal.

Source: https://www.baeldung.com/wp-content/uploads/sites/4/2022/02/img_6210aec6c9d02.svg

6.6.5.4. Besoin élevé en ressources

Si le besoin en RAM GPU est élevé pour stocker les paramètres d'un LLM, il est environ six fois plus important lorsqu'il s'agit de les fine-tuner. Le besoin computationnel reste le plus gros défi pour les structures disposant d'un budget restreint.

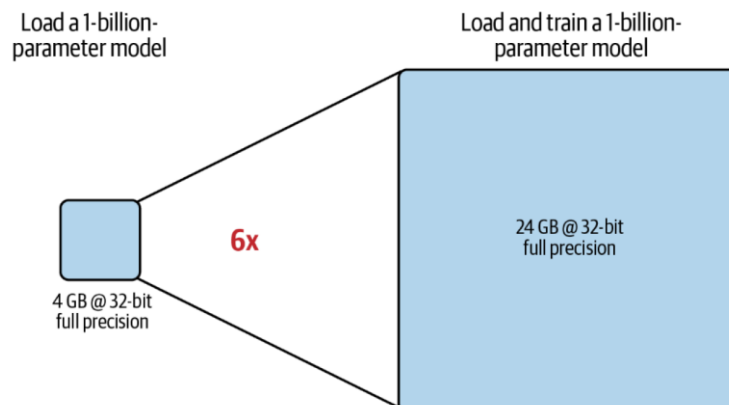


Figure 15: Besoin en RAM GPU pour charger et pour entraîner un modèle.
Source : <https://vitalflux.com/wp-content/uploads/2024/01/LLM-GPU-Memory-Requirements-300x170.png>

Il existe néanmoins des mécanismes de quantification qui permettent de réduire la taille du modèle de 32 à 16, 8 ou 4bits et ainsi réduire la mémoire nécessaire sans péjorer de manière trop significative sa précision.

6.6.6. Choix final

La stratégie de fine-tuning retenue pour la réalisation de ce projet est la méthode **LoRA**. Cette méthode permet d'approcher les performances du *full parameter fine-tuning* beaucoup plus rapidement et avec de faibles ressources matérielles.

6.7. Gestion des données

Dans le cadre de ce projet, les données qui nécessitent d'être enregistrées et accessibles sont restreintes. En effet, seuls les identifiants des utilisateurs ainsi que les outputs des requêtes envoyées au LLM sont enregistrés. Une base de données relationnelle est donc parfaite pour ce type de traitement.

Etant donné que les différents composants de mon projet sont hébergés par l'IT-HVS, nous nous sommes intéressés aux systèmes de gestion de base de données relationnelles qu'il préconisait. Le système qu'il nous a conseillé est **PostgreSQL**

PostgreSQL est un système de base de données relationnelle et objet open-source qui convient aux entreprises de toute taille. Robuste, fiable et gratuit, PostgreSQL est l'un des systèmes les plus utilisés au monde. Il est facilement extensible, ce qui constitue un atout

important pour les développements futurs du projet. De plus, il dispose de bonnes fonctionnalités de sécurité comme l'authentification SSL ou le chiffrement des données.

7. Risques du projet

Cette section liste les différents risques qui découlent de l'utilisation de LLM au sein d'une entreprise. Ces risques ont été cités et expliqués en détails lors d'une conférence organisée par CLUSIS, référence suisse dans le domaine de la sécurité de l'information, et prodiguée par l'entreprise Wavestone, spécialisée dans le conseil en transformation stratégique et l'intégration de l'IA générative dans les entreprises. Cette conférence a eu lieu en ligne le 30 avril 2024 (Wavestone, CLUSIS, 2024).

7.1. Biais

Les LLM sont des outils puissants mais qui peuvent comporter des risques de biais qui peuvent avoir des conséquences importantes. Ces biais proviennent généralement des données d'entraînement et des interactions des utilisateurs, et sont reproduits par le modèle lors de la génération de texte. Ces biais peuvent renforcer les stéréotypes, entraîner des décisions injustes, et poser des problèmes éthiques et juridiques.

7.2. Cyber-attaques répertoriées

Les LLM ont un potentiel énorme pour les entreprises. Malheureusement, comme tout système, l'IA contient des failles qui peuvent être exploitées par des attaquants. Voici quelques cyber-attaques connues sur les systèmes IA.

7.2.1. Attaque par empoisonnement

Les attaques par empoisonnement sont un type d'attaque qui cible directement les datasets entraînés qui sont utilisés par les IA ou les LLM. L'objectif est d'influencer ou manipuler les opérations réalisées par ces modèles.

Il existe plusieurs manières de faire du data poisoning :

- En injectant intentionnellement des données erronées directement dans le dataset entraîné.
- En modifiant des données déjà existantes dans le dataset entraîné.
- En supprimant une partie des données du dataset.

En manipulant les datasets, un attaquant peut introduire des biais, provoquer des outputs erronés ou influencer la prise de décision et la capacité de prédiction du modèle.

7.2.2. Attaque Oracle

Un oracle est un type d'attaque où un adversaire cherche à manipuler un LLM à travers du prompt engineering, dans le but d'extraire des informations que celui-ci ne devrait pas révéler.

Déroulement d'une attaque Oracle :

1. Interrogation du modèle : L'attaquant fait du prompt engineering sur le modèle pour que celui-ci lui révèle progressivement des informations sur sa structure interne et sur la manière avec laquelle il a été entraîné.
2. Analyse des réponses : Grâce à l'analyse des réponses obtenues lors de la phase 1, l'attaquant va tenter de déceler des patterns ou des lacunes exploitables sur le modèle pour lui soutirer encore plus d'informations.
3. Extraction d'informations : À travers cette méthode itérative, un attaquant peut parvenir à reconstruire des informations sur les données d'entraînement (comme extraire des exemples spécifiques de données), ou découvrir des détails sur l'architecture du modèle, comme sa façon de pondérer certains types d'entrées.

7.2.3. Attaque par manipulation

Les attaques par manipulation sont des techniques utilisées pour tromper, perturber ou manipuler le fonctionnement d'un système d'IA. Ce type d'attaque s'effectue sur les données d'entrées (données d'input) et vise à exploiter les vulnérabilités des LLM dans la manière dont il traite et interprète les inputs.

Types de manipulation attacks :

- Attaque par perturbation : L'attaquant va apporter des modifications subtiles à des images, des sons ou d'autres types d'entrées afin que l'IA soit incapable de les interpréter correctement. Comme ces modifications sont légères, elles sont indétectables par un humain mais faussent l'interprétation effectuée par l'IA. Par exemple, une légère modification des pixels peut pousser une IA de reconnaissance d'image à classer un panneau « STOP » comme un panneau « Cédez le passage ».

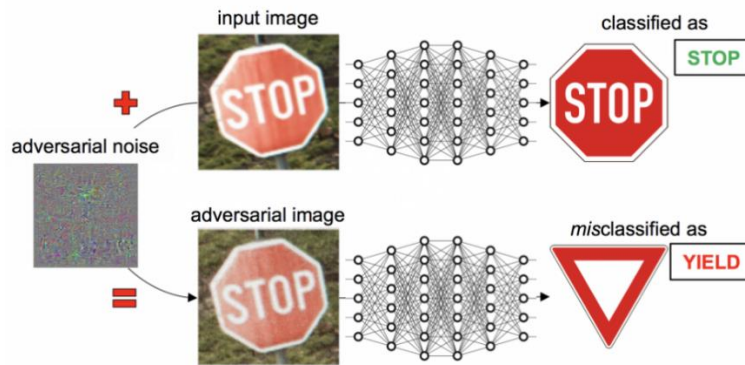


Figure 16: Exemple d'attaque par perturbation

Source: <https://medium.com/@souravagarwal54321/machine-learning-attack-a92c5359b36d>

- Attaque de contournement : L'attaquant va modifier des données d'entrée pour contourner les systèmes de sécurité basés sur l'IA, comme les systèmes de reconnaissance faciale ou de détection de fraude.

7.3. Hallucinations

Il est nécessaire de vérifier les réponses générées par un LLM, car celles-ci peuvent parfois contenir des erreurs. En effet, les modèles reproduisent les connaissances acquises avec les données d'entraînement. Si ces données contiennent des erreurs, elles peuvent se reporter dans les réponses générées sous forme d'hallucinations (génération d'informations infondées qui semblent plausibles mais sont en réalité fictives), d'incohérences ou de réponses hors-sujet par manque de contexte. De plus, il est important de garder en tête qu'un LLM ne dira jamais qu'il ne sait pas, mais générera une réponse fictive s'il ne possède pas l'information.

8. Développements

Cette section présente et explique les différentes implémentations réalisées durant le projet. Pour les lecteurs souhaitant obtenir les détails des étapes d'installation de l'environnement, du déploiement et de certaines phases de fine-tuning, un guide technique est disponible en annexe (Annexe 5). De plus, l'intégralité du code développé est disponible sur un répertoire GitHub (Annexe 3).

8.1. Structuration des données d'entraînement

Comme expliqué au point 5.1.1.2, dans un but de garantie de conformité à la nLPD, les données d'entraînement utilisées dans le cadre de ce travail, proviennent d'un *dataset*

disponible en open-source sur la plateforme Hugging Face. Il s'agit d'un fichier JSON de 10'000 entrées qui regroupe des définitions de pathologies, leurs symptômes et les traitements possibles. Ces données proviennent de Wikidoc, une plateforme collaborative dédiée à la documentation médicale et alimentée par des professionnels de la santé.

Le *dataset* utilisé est au format Alpaca. Le format Alpaca est un format structuré, utilisé pour entraîner des LLM sur des tâches spécifiques en fournissant des exemples clairs d'instructions, d'entrées, et de sorties attendues.

input string · lengths	output string · lengths	instruction string · classes
60-90 40.9%	2-6.89k 98.9%	Answer thi... 100%
Can you provide an overview of the lung's squamous cell carcinoma?	Squamous cell carcinoma of the lung may be classified according to the WHO histological classification system into 4 main types: papillary, clear cell, small cell, and basaloid.	Answer this question truthfully
What does "Clear: cell" mean?	Clear cell tumors are part of the surface epithelial-stromal tumor group of Ovarian cancers...	Answer this question truthfully
Can you provide me with information regarding statins?	Two Japanese scientists commenced research into inhibitors of HMG-CoA reductase in 1971 reasoning...	Answer this question truthfully
What are the historical background and symptoms of Candida-induced vulvovaginitis?	Symptoms of vulvovaginitis caused by Candida species are indistinguishable and include the...	Answer this question truthfully
What does the "Hypotension: Resident Survival Guide" refer to?	Hypotension is the term for low blood pressure (BP). A systolic BP measuring less than 90mmHg...	Answer this question truthfully

Figure 17: Echantillon représentant les 5 premières entrées du fichier json.
Source de l'auteur

8.2. Conception des prompts

Le développement de ce projet a nécessité la conception de deux prompts, un pour chaque *use case* implémenté.

8.2.1. Prompt de vulgarisation et traduction

Ce prompt a pour but de vulgariser et traduire un texte médical fourni par l'utilisateur afin de le rendre compréhensible par le patient.

Le prompt est construit sous cette forme :

```
prompt = (
    f"""You are a healthcare professional (nurse or care assistant) and
    you must answer in {language}.
```

Vulgarize the text below in simple words that a 10-year-old child or an elderly person can understand. Start this section with 'Simplification: '.

If the text contains medical terms, organs, or body parts, explain these words in simple words that everyone can understand. Start this section with 'Termes: '.

Finally, give a summarized name in 5 words to this conversation, and start it with 'Summary: '.

Do not include any other text or symbols like ''. Here is the text: ""*

+ text

)

Le prompt est divisé en 5 parties.

1. En **VERT**, c'est le contexte qui est donné.
2. En **BLEU**, c'est l'instruction de vulgarisation.
3. En **ROSE**, c'est la fonctionnalité d'explication des termes complexes.
4. En **ORANGE**, c'est la fonctionnalité de résumé en 5 mots qui permet d'identifier le prompt dans l'historique de l'application.
5. En **GRIS**, ce sont les consignes pour éviter la génération de texte ou symbole inutile qui pourrait perturber le traitement de la réponse.

Deux paramètres récupérés depuis le *front end* sont passés directement dans le prompt :

1. **language** est la langue dans laquelle l'utilisateur souhaite que la génération soit faite
2. **text** est le texte médical fourni par l'utilisateur

8.2.2. Prompt de résumé du dossier patient

Ce prompt a pour but de résumer un dossier patient fourni sous forme de données structurées au format XML.

Le prompt est construit sous cette forme :

```
prompt = (
f""You are a text summarization and medical expert. Given the
following medical record, generate a medical summary in {language}
in the following format (replace the values between "" by what you
found in the medical record):
Record: {combined_text}
Format:
```

Données personnelles:
- Prénom: "Prénom"
- Nom : "Nom"
- Sexe: "Sexe"
- N°AVS: "N°AVS"
- Date de naissance: "Date de naissance"
- Adresse: "Adresse"

Cas:
"*Complete sentence describing the case*"

Réseau social:
- Référent: "Référent", tel: "Tel référent"
- Médecin traitant: "Nom du médecin traitant", tel: "Tel médecin"

Informations générales:
"*Complete sentence*"

Allergies:
- "*List of allergies*"

Historique médical:
"*Complete sentence describing the "Diagnostics"*"

Médication:
- Voie orale:
- "*List of médicaments*"
- Injection:
- "*List of médicaments*"
""
)

Le prompt est divisé en 4 parties :

1. En **VERT**, c'est le contexte qui est donné.
2. En **BLEU**, c'est l'instruction du résumé du dossier.
3. En **ROSE**, c'est la définition du dossier à résumer.
4. En **ORANGE**, c'est le format souhaité pour la réponse du LLM. Il regroupe toutes les informations essentielles qui doivent figurer dans le résumé.

Deux paramètres récupérés depuis le *front end* sont passés directement dans le prompt :

1. **language** est la langue dans laquelle l'utilisateur souhaite que la génération soit faite.

2. **combined_text** est le texte extrait du fichier XML uploadé par l'utilisateur et qui doit être résumé.

8.3. Entraînement du LLM et intégration

L'entraînement du LLM Llama3 8B et l'intégration du nouveau modèle sur le serveur LLM Ollama nécessitent plusieurs étapes et l'utilisations d'outils spécialisés.

8.3.1. Outils

Hugging Face

Hugging Face est une plateforme spécialisée dans le développement et la diffusion de modèles NLP. Elle contient de nombreux modèles de base et modèles fine-tunés open-source ainsi que des *datasets* préparés pour le fine-tuning. C'est sur cette plateforme que nous avons récupéré le modèle pré-entraîné Llama3 8B utilisé comme base pour le fine-tuning.

Google Colab

Google Colab est un environnement de développement cloud qui permet d'exécuter des commandes python directement sur leur navigateur. Il est également possible de louer de la puissance de calcul à des prix raisonnables, ce qui est particulièrement intéressant pour réaliser des tâches de fine-tuning. Nous avons utilisé Google Colab pour réaliser le fine-tuning de mon modèle pré-entraîné Llama3 8B en souscrivant à un abonnement nous permettant d'utiliser des GPU puissants (Nvidia A100).

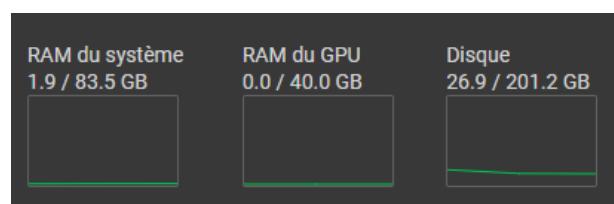


Figure 18: Ressources à disposition sur Google Colab
Source de l'auteur

8.3.2. Entraînement

L'entraînement du modèle a été réalisé sur Google Colab à l'aide d'un *template* créé par Unsloth et spécialement préparé pour entraîner des LLM (<https://github.com/unslothai/unsloth.git>). Des modifications ont été apportées à ce *template*

pour répondre aux besoins spécifiques de ce projet. L'ensemble du code exécuté sur l'environnement Google Colab est disponible sur le git du projet, dans le répertoire `optirh_finetuning/training_steps`.

L'entraînement se déroule en plusieurs étapes successives qui peuvent être décomposées. Dans chacune de ces étapes, plusieurs paramètres sont à configurer suivant l'objectif du fine-tuning, le type de modèle, le dataset utilisé et la puissance de calcul à disposition (Labonne, 2024). Ces étapes sont décrites et expliquées dans les chapitres suivants.

8.3.2.1. Chargement du modèle pré-entraîné

Cette étape consiste à charger un modèle pré-entraîné et son tokenizer, puis à préparer le modèle pour l'utiliser lors de la phase de fine-tuning.

```
model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/llama-3-8b-bnb-4bit",
    max_seq_length = 2048,
    dtype = None,
    load_in_4bit = True,
    token = "hf_D0rQRBDMeMoLzjJrSecFHogfuMXODTBEBH",
```

Figure 19: Chargement du modèle pré-entraîné
Source de l'auteur

Voici une explication des paramètres :

model_name	Détermine le modèle pré-entraîné à charger sur l'environnement.
max_seq_length	Détermine la longueur maximale de la séquence que le modèle peut traiter.
dtype	Spécifie le type de données (dtype) utilisé pour les poids du modèle.
load_in_4bit	Indique que le modèle doit être chargé en utilisant une représentation en 4bit pour les poids, ce qui permet de réduire la mémoire nécessaire pour le stockage du modèle.
token	Si le modèle de base est chargé depuis un repo Hugging Face nécessitant un droit d'accès, il est nécessaire de préciser le token de son compte.

Justification des choix :

- **model_name** : Le modèle *unsloth/llama-3-8b-bnb-4bit* est un modèle Llama 3 8B quantifié en 4bit. Cela permet un téléchargement plus rapide du modèle sur l'environnement Colab.
- **max_seq_length=2048** : La longueur maximale de séquence de 2048 tokens est un standard pour les LLM. Cela permet de générer des contextes relativement long.
- **Dtype=None** : La valeur *None* permet au modèle de choisir automatiquement le type de données optimale par rapport au GPU utilisé pour le fine-tuning.
- **load_in_4bit=True** : Le chargement du modèle en 4 bits permet de réduire l'utilisation de la mémoire GPU nécessaire pour l'entraînement. Cette configuration permet de maintenir une précision acceptable sur des ressources limitées.

8.3.2.2. Préparation du modèle pour un fine-tuning avec LoRA

Comme expliqué au point 6.6.4, LoRA permet de réduire significativement le nombre de paramètres entraînés, ce qui réduit le besoin de mémoire et augmente la rapidité de l'entraînement. Il convient cependant de bien préparer sa méthode LoRA avant de commencer le fine-tuning.

```
[3] model = FastLanguageModel.get_peft_model(
    model,
    r = 16, # Choose any number > 0 ! Suggested 8, 16, 32, 64, 128
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                     "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 16,
    lora_dropout = 0, # Supports any, but = 0 is optimized
    bias = "none",    # Supports any, but = "none" is optimized
    # [NEW] "unsloth" uses 30% less VRAM, fits 2x larger batch sizes!
    use_gradient_checkpointing = "unsloth", # True or "unsloth" for very long context
    random_state = 3407,
    use_rslora = False, # We support rank stabilized LoRA
    loftq_config = None, # And LoftQ
)
```

Figure 20: Paramétrage de LoRA
 Source de l'auteur

Voici une explication des paramètres les plus importants :

r (Rank)	Détermine la taille des matrices utilisées dans LoRA. Un rang plus élevé permet une meilleure capacité d'adaptation, mais augmente le besoin en puissance de calcul.
target_modules	Spécifie quelles couches du modèle seront modifiées pendant le fine-tuning.
lora_alpha	Détermine l'échelle des poids LoRA par rapport aux poids originaux. Un facteur plus élevé augmente l'impact des ajustements LoRA. En général, ce paramètre est égal ou le double de r.

Justification des choix :

- **r=16** : Une valeur 16 permet une bonne flexibilité pour ajuster les poids du modèle tout en garantissant un besoin raisonnable en puissance de calcul.
- **target_modules** : Les couches définies dans cette variable sont cruciales pour le fonctionnement du modèles. Ce sont également les couches les plus adeptes à être fine-tunées. Cette configuration est courante pour les entraînements de type LoRA.
- **lora_alpha=16** : le rang ayant été fixé à 16, nous avons repris la même valeur pour lora_apha. Cette configuration offre un bon compromis entre la régularisation et la capacité d'apprentissage du modèle.

Cette configuration nous permet de concentrer l'entraînement sur 42 millions de paramètres sur les 8 milliards qui composent le modèle pré-entraîné, tout en gardant une performance proche du full fine-tuning.

8.3.2.3. Préparation des données d'entraînement

Une fois le modèle pré-entraîné chargé, il faut préparer les données qui seront utilisées pour l'entraînement. Les *datasets* d'instruction peuvent être de plusieurs formats. Dans notre cas, il s'agit d'un format Alpaca (instruction, input, output). L'objectif de cette étape est de transformer les données brutes en un format compatible avec notre modèle. Le but est d'inclure des instructions, des entrées contextuelles et des réponses de manière structurée, en ajoutant un token de fin de séquence pour éviter des générations infinies.

```

from datasets import load_dataset, Dataset
from sklearn.model_selection import train_test_split

dataset = load_dataset("medalpaca/medical_meadow_wikidoc", split="train")

train_dataset_dict, val_dataset_dict = train_test_split(dataset, test_size=0.2, random_state=42)

train_dataset = Dataset.from_dict(train_dataset_dict)
val_dataset = Dataset.from_dict(val_dataset_dict)

alpaca_prompt = """Below is an instruction that describes a task, paired with an input that provides further context.
Write a response that appropriately completes the request.

### Instruction:
{}

### Input:
{}

### Response:
{}"""

EOS_TOKEN = tokenizer.eos_token

def formatting_prompts_func(examples):
    instructions = examples["instruction"]
    inputs = examples["input"]
    outputs = examples["output"]
    texts = []
    for instruction, input, output in zip(instructions, inputs, outputs):
        # Must add EOS_TOKEN, otherwise your generation will go on forever!
        text = alpaca_prompt.format(instruction, input, output) + EOS_TOKEN
        texts.append(text)
    return {"text": texts}

# Formater les ensembles de formation et de validation
train_dataset = train_dataset.map(formatting_prompts_func, batched=True)
val_dataset = val_dataset.map(formatting_prompts_func, batched=True)

# Optionnel: Prévisualiser quelques exemples formatés
print(train_dataset['text'][:3])
print(val_dataset['text'][:3])

```

Figure 21: Préparation du dataset d'entraînement
Source de l'auteur

Les données d'entraînement ont été séparées en deux ensembles : un ensemble d'entraînement (80% des données) et un ensemble de validation (20% des données). Cette démarche permet de vérifier la qualité d'apprentissage, en évaluant la performance du modèle sur des données non vues pendant l'entraînement. Cette vérification se fait en analysant les logs des métriques de *training loss* et de *validation loss*.

8.3.2.4. Définition des hyperparamètres d'entraînement

Les hyperparamètres définissent la configuration ou les réglages du modèle. Ils sont définis avant l'entraînement d'un modèle et ne sont pas appris directement à partir des données. Contrairement aux paramètres du modèle (comme les poids) qui sont ajustés pendant l'entraînement, les hyperparamètres influencent le processus d'entraînement et la structure du modèle.

SFTTrainer (Supervised Fine-Tuning Trainer) est un outil utilisé dans le cadre du fine-tuning supervisé des modèles de langage. Il prend, en paramètres, toutes les configurations d'hyperparamètres de notre entraînement.

```

from trl import SFTTrainer
from transformers import TrainingArguments
from unsloth import is_bfloat16_supported

# Configuration du trainer
trainer = SFTTrainer(
    model=model, # Modèle pré-entraîné à fine-tuner.
    tokenizer=tokenizer, # Tokenizer associé au modèle.
    train_dataset=train_dataset, # Dataset d'entraînement.
    eval_dataset=val_dataset, # Dataset de validation.
    dataset_text_field="text", # Champ texte dans le dataset.
    max_seq_length=max_seq_length, # Longueur max des séquences.
    dataset_num_proc=2, # Nombre de processus pour le dataset.
    packing=False, # Regroupement des séquences (False).
    args=TrainingArguments(
        per_device_train_batch_size=8, # Taille du batch d'entraînement.
        per_device_eval_batch_size=8, # Taille du batch d'évaluation.
        gradient_accumulation_steps=2, # Accumulation de gradients.
        warmup_steps=5, # Étapes de warmup.
        max_steps=-1, # Pas de limite sur les étapes.
        num_train_epochs=1, # Nombre d'époques.
        learning_rate=2e-4, # Taux d'apprentissage.
        fp16=not is_bfloat16_supported(), # Utilisation du float16.
        bf16=is_bfloat16_supported(), # Utilisation du bfloat16.
        logging_steps=10, # Fréquence des logs.
        evaluation_strategy="steps", # Stratégie d'évaluation par étapes.
        eval_steps=10, # Fréquence des évaluations.
        save_steps=100, # Fréquence des sauvegardes.
        save_total_limit=20, # Limite de sauvegardes.
        optim="adamw_8bit", # Optimiseur AdamW 8-bit.
        weight_decay=0.01, # Décroissance des poids.
        lr_scheduler_type="linear", # Scheduler linéaire.
        seed=3407, # Graine pour la reproductibilité.
        output_dir="outputs", # Dossier de sortie.
        load_best_model_at_end=True, # Charger le meilleur modèle à la fin.
        metric_for_best_model="eval_loss", # Métrique pour le meilleur modèle.
        greater_is_better=False, # Moins de perte est meilleur.
    )
),

```

Figure 22: Hyperparamètres passés au SFTTrainer
Source de l'auteur

Voici une explication des paramètres les plus importants :

per_device_batch_size	Détermine la taille du batch traité par chaque GPU. La taille du batch est le nombre d'échantillons du dataset traités avant la mise à jour des poids du modèle. Des batchs plus grands améliorent généralement la stabilité et la vitesse de l'entraînement, mais ils nécessitent également plus de mémoire GPU.
gradient_accumulation_steps	Ce paramètre est particulièrement utilisé quand la mémoire GPU est limitée. Il permet de diviser un batch effectif en

	plusieurs sous-batches plus petit. La mise à jour des poids du modèle se fait seulement une fois que ces sous-batches sont accumulés.
warmup_steps	Nombre de pas d'entraînement initial pendant lesquels le taux d'apprentissage augmente linéairement de zéro jusqu'à la valeur spécifiée. La phase de warmup aide généralement à stabiliser le début de l'entraînement.
num_train_epochs	Nombre d'époques d'entraînement, où une époque est un passage complet à travers l'ensemble des données d'entraînement. Voir plusieurs fois l'ensemble du modèle peut permettre au modèle de mieux assimiler les données mais peut aussi causer un surentraînement.
learning_rate	Contrôle l'intensité avec laquelle le modèle met à jour ses paramètres. S'il est trop élevé, l'entraînement pourrait devenir instable ou diverger. S'il est trop bas, il sera lent.
logging_step	Fréquence (en nombre d'étapes) à laquelle les logs sont enregistrés pour surveiller l'entraînement.
optim	L'optimiseur utilisé pour ajuster les poids du modèle.
weight_decay	Une technique de régularisation qui pénalise les poids élevés dans la fonction de perte. Elle prévient le surapprentissage en favorisant des caractéristiques plus simples et généralisables. Une pénalisation excessive peut cependant nuire à l'apprentissage.
lr_scheduler_type	Type de planificateur de taux d'apprentissage qui ajuste le taux d'apprentissage au cours de l'entraînement. Les planificateurs linéaires et cosinus sont couramment utilisés.
load_best_model_at_end	Permet de charger le meilleur modèle enregistré lors de l'entraînement quand le processus est terminé. La métrique

	utilisée pour l'évaluation du meilleur modèle doit être précisée.
--	---

Justification des choix :

- **per_device_train_batch_size=8** : Une taille de 8 est un compromis entre la stabilité de l'entraînement et les limitations de mémoire. Cela permet de traiter un nombre d'exemples suffisamment grand pour que le modèle apprenne efficacement tout en respectant la capacité mémoire du GPU.
- **Gradient_accumulation_steps=2** : Disposant d'une mémoire limitée, ce paramétrage permet de simuler un batch size de 16 pour augmenter la performance du modèle sans nécessiter plus de mémoire GPU.
- **warmup_steps=5** : Le *dataset* utilisé étant relativement petit, 5 étapes pourraient suffire pour éviter les fluctuations au début de l'entraînement.
- **num_train_epochs=1** : Plusieurs entraînements, avec des nombre d'époques différents, ont été réalisés. La validation croisée des métriques de *training* et validation loss a montré que le modèle était victime de surentraînement dès la deuxième époque. Un entraînement sur une époque a donc été jugé optimal.
- **Learning_rate=2e-4**: Un taux de 2e-4 est souvent utilisé pour du fine-tuning, car il permet d'apporter des ajustements fins aux poids du modèle sans déstabiliser les apprentissages précédents.
- **logging_steps=10** : Une fréquence de log chaque 10 étapes permet d'avoir un suivi précis de la progression, sans générer un volume de logs excessif.
- **Optim=damw_8bit** : Damw_8bit est une version optimisée de l'optimiseur Adam utilisant des poids en 8 bits pour réduire la mémoire utilisée. Cet optimisateur maintient des performances robustes tout en limitant l'utilisation de la RAM GPU.
- **Weight_decay=0.01** : Une faible régularisation de 0.01 permet de réduire les risques de surapprentissage en pénalisant les grands poids.
- **lr_scheduler_type=linear** : Un *scheduler* linéaire réduit progressivement le taux d'apprentissage et permet une convergence stable. Ce type de *scheduler* est particulièrement utilisé pour le fine-tuning de modèle pré-entraîné.

- **Load_best_model_at_end=True:** Ce paramètre permet de charger le meilleur modèle sauvegardé lors de l'entraînement. Cette fonction est notamment utile lorsqu'un nombre d'*epochs* élevé est choisi et que le risque de surapprentissage est grand. Dans notre cas, la métrique choisie pour l'évaluation du meilleur modèle est la *validation loss*.

Une fois ces paramètres définis, l'entraînement du modèle pré-entraîné peut débuter.

8.3.2.5. Analyse des logs d'entraînement

Une fois l'entraînement terminé, il est vital d'analyser les logs pour s'assurer que le modèle a été entraîné de manière optimale. La validation croisée des *validation et training loss*, nous permet de vérifier que notre modèle n'est pas victime de surentraînement ou de sous-entraînement.

De manière graphique, les courbes de *training loss* et de *validation loss* de notre entraînement se présente comme suit :

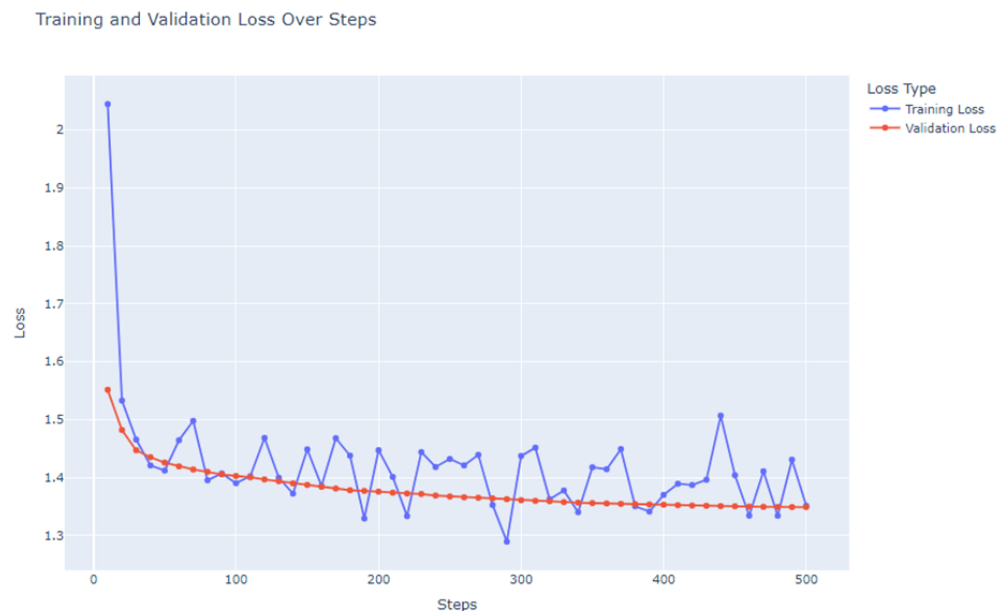


Figure 23: Courbes de training et validation loss à travers les étapes.
Source de l'auteur

Ce graphique montre, globalement, une diminution des pertes au fur et à mesure des étapes. Cela indique que l'apprentissage est efficace. Les pertes importantes dans les premières étapes suggèrent que le nombre de *warmup steps* est un peu insuffisant. La proximité entre les deux types de pertes suggère une bonne généralisation. La convergence des pertes, vers la fin, indique que le modèle atteint un plateau de performance. L'absence de divergence significative entre les pertes suggère qu'il n'y a pas de surentraînement. En résumé, le modèle est bien équilibré et ne montre pas de signes de surentraînement ou de sous-entraînement.

8.3.3. Fusion du modèle et de l'adapter

Lorsque la stratégie LoRA est utilisée, le résultat obtenu après l'entraînement n'est pas un nouveau modèle complet mais un *adapter*. Cet *adapter* contient les matrices de rang faible A et B qui regroupent les nouveaux paramètres appris durant le fine-tuning grâce au dataset d'entraînement. La matrice A capture les caractéristiques essentielles de la tâche spécifique pour laquelle le modèle a été fine-tuné, alors que la matrice B capture comment les caractéristiques essentielles devraient être combinées pour produire les sorties appropriées. Seul, cet adapter n'est pas d'une grande utilité. Pour l'utiliser, il faut le recharger dans le modèle qui a été utilisé comme base pour le fine-tuning.

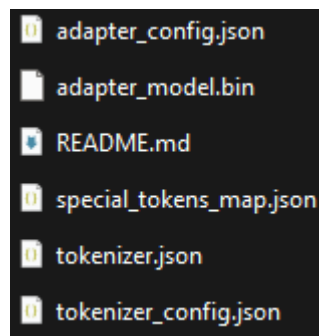


Figure 24: Fichiers composant l'adapter LoRA
 Source de l'auteur

- **adapter_config.json** : Ce fichier contient les configurations et les paramètres spécifiques pour l'adapter LoRA. Il définit comment les matrices à faible rang (A et B) doivent être appliquées aux poids du modèle de base.
- **adapter_model.bin** : Ce fichier binaire contient les poids des matrices à faible rang (A et B) et potentiellement d'autres paramètres de l'adapter.

- **README.md** : Ce fichier Markdown contient des informations et des instructions sur l'utilisation de l'adapter
- **special_tokens_map.json** : Ce fichier mappe des tokens spéciaux utilisés par le tokenizer, tels que les tokens de début, de fin, de padding, et autres.
- **tokenizer.json** : Ce fichier contient la configuration et les paramètres du tokenizer utilisé pour le modèle.
- **tokenizer_config.json** : Ce fichier spécifie des configurations supplémentaires pour le tokenizer, incluant des paramètres sur la manière dont il doit être initialisé et utilisé.

Afin de fusionner l'*adapter* et le modèle de base, et de les uploader sur notre compte personnel sur la plateforme Hugging Face, nous avons créé deux classes python :

- **merge_models.py** : classe qui fusionne le modèle de base et l'adapter
- **upload_model_on_hf.py**: classe qui upload le modèle sur Hugging Face

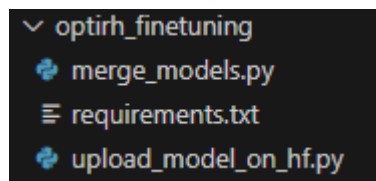


Figure 25: classes python pour la fusion de l'adapter et du modèle et l'upload sur Hugging Face
Source de l'auteur

Une fois la fusion effectuée, le modèle qui en découle est complet, pleinement utilisable et entraîné avec les données du dataset. Les modèles fine-tunés sont stockés sur la plateforme Hugging Face (Annexe 4).

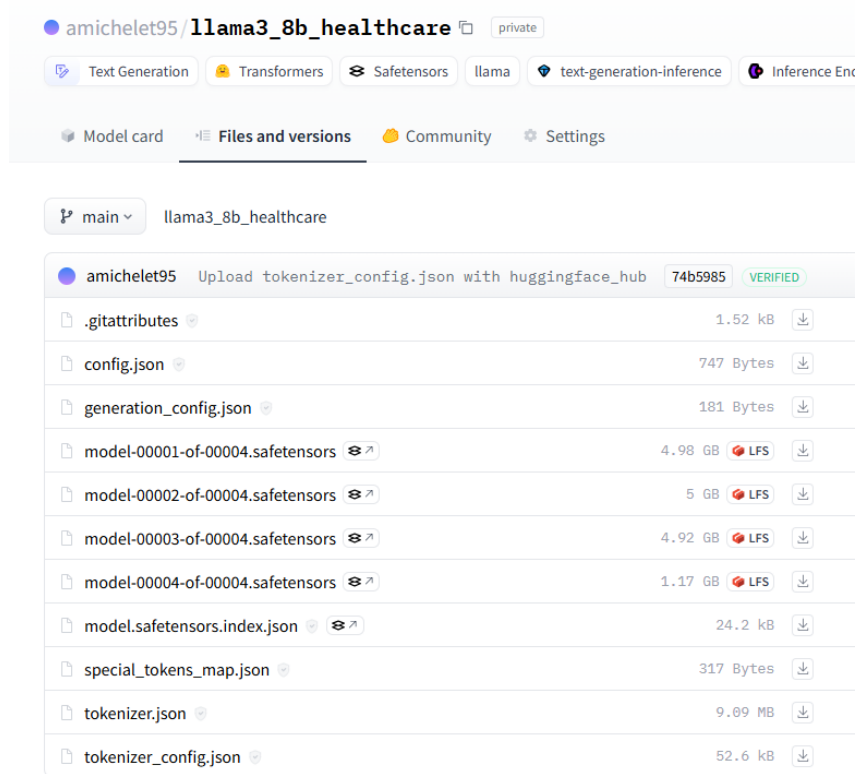


Figure 26: Modèle final fine-tuné et disponible sur Hugging Face Source de l'auteur

8.3.4. Conversion du modèle en .gguf

Si le modèle fine-tuné qui découle de la fusion entre le modèle pré-entraîné et l'*adapter* est utilisable en l'état, il ne peut cependant pas être intégré à l'outil Ollama qui se charge de l'exécution du modèle dans ce projet. En effet, Ollama n'accepte d'intégrer que les modèles au format .gguf. Ce format permet de réduire la taille des modèles en utilisant des techniques de quantification, ce qui rend les modèles plus efficaces en termes de stockage et de calcul. Il est donc nécessaire de convertir notre modèle dans ce format.

Pour réaliser cette opération, nous avons utilisé le repository git <https://github.com/ggerganov/llama.cpp.git> qui propose plusieurs outils de conversion de LLM. Les détails techniques pour la réalisation de cette étape sont documentés dans le Guide technique en annexe (Annexe 5).

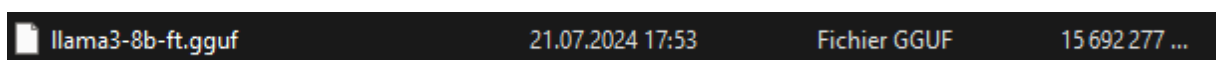


Figure 27: Modèle fine-tuné au format gguf

8.3.5. Création du nouveau model sur Ollama

Finalement, le modèle converti au format .gguf, doit être chargé sur Ollama. Pour cela, il est nécessaire de créer un fichier nommé « Modelfile ».

Ce fichier « Modelfile » contient trois informations importantes pour Ollama :

```
FROM C:/PERSO/TB/Models/gguf/llama3-8b-ft.gguf

TEMPLATE """Below are some instructions that describe some
tasks. Write responses that appropriately complete each request.{{ if
.Prompt }}
### Instruction:
{{ .Prompt }}
{{ end }}### Response:
{{ .Response }}<|end_of_text|>"""

PARAMETER stop "<|start_header_id|>"
PARAMETER stop "<|eot_id|>"
PARAMETER stop "<|end_header_id|>"
PARAMETER stop "<|end_of_text|>"
PARAMETER stop "<|reserved_special_token_"
```

1. En **VERT**, c'est le chemin système vers le fichier .gguf à importer.
2. En **BLEU**, c'est le *template* qui détermine sous quel format les prompts et les réponses seront construits lors de l'utilisation du modèle.
3. En **ORANGE**, ce sont les paramètres d'exécution.

Une fois le « Modelfile » défini, il faut exécuter la commande :

```
ollama create nom_model -f Modelfile
```

Le modèle est maintenant créé dans Ollama sous le nom donné.

```
C:\Users\Win 11>ollama run llama3hc
>>> Send a message (? for help)
```

Figure 28: LLM fine-tuné chargé sur Ollama
Source de l'auteur

8.4. Back-end : API Django

L'API Django est le point central de l'application de démonstration. Elle gère les requêtes de l'application frontend en assurant une communication fluide avec le serveur Ollama et la base de données. L'application frontend envoie des requêtes HTTP que l'API Django traite via des vues appropriées, renvoyant les réponses nécessaires. L'API se charge également de l'envoi des prompts destinés au LLM fine-tuné vers le serveur Ollama. Les réponses d'Ollama sont analysées et formatées par l'API Django pour être utilisées par le frontend ou stockées dans la base de données. Pour la partie CRUD, l'API utilise l'ORM de Django pour écrire et récupérer des données, assurant ainsi l'intégrité et la cohérence des informations. En résumé, Django assure une interaction efficace entre le frontend, Ollama et la base de données, garantissant des performances optimales et une gestion cohérente des données.

8.4.1. Structure du projet Django

La structure du projet Django est construite comme suit :

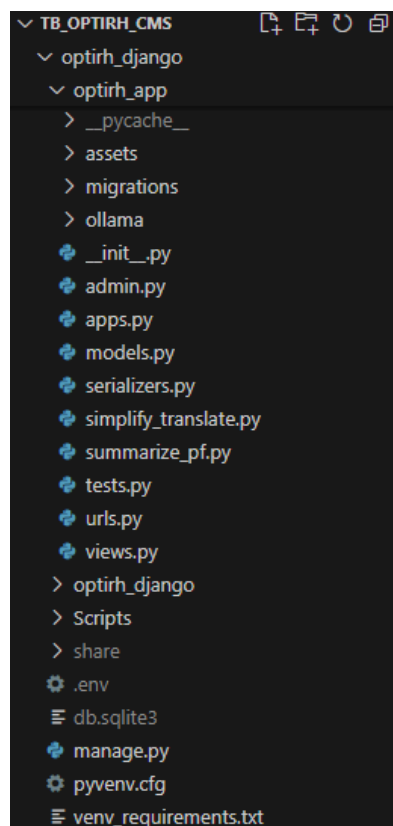










Figure 29: Structure du projet Django
Source de l'auteur

 migrations	Ce dossier contient toutes les migrations de la base de données. Les fichiers de ce dossier sont générés automatiquement.
 ollama	Ce dossier contient la classe utilisée pour gérer l'appel à Ollama. C'est ici que l'on définit notamment le LLM utilisé.
 models.py	Ce fichier python contient tous les modèles de données de l'application. C'est à partir de ces modèles que sont créées les tables de la DB.
 serializers.py	Ce fichier python contient les <i>serializers</i> qui s'occupent de convertir les objets en chaîne de caractère json afin de pouvoir les envoyer dans des réponses d'API, et inversement.
 urls.py	Ce fichier python contient les méthodes de l'API django. C'est donc la liste des méthodes que l'on peut appeler depuis le frontend.
 views.py	Ce fichier python contient les vues qui définissent la logique de traitement des requêtes provenant de l'interface utilisateur.
 simplify_translate.py	Ce fichier python contient toutes les méthodes nécessaires au traitement du use case de communication patient-soignant.
 summarize_pf.py	Ce fichier python contient toutes les méthodes nécessaires au traitement du use case de résumé du dossier patient.

8.4.2. Sécurité

Dans l'objectif d'accroître la sécurité, des mécanismes d'authentification par token et de permissions ont été mis en place au sein de l'application Django. Combinés, ils assurent que seules les requêtes provenant d'utilisateurs authentifiés et autorisés accèdent aux vues.

Au niveau du fonctionnement, lors de la connexion, l'utilisateur reçoit un jeton d'authentification unique qu'il doit inclure dans l'en-tête de la requête HTTP afin d'accéder à des vues protégées. La classe « TokenAuthentication » extrait ce jeton, le vérifie et identifie

l'utilisateur associé. Ensuite, la classe « IsAuthenticated » permet l'accès à la vue aux utilisateurs ayant réussi l'authentification.

```
@api_view(['GET'])
@authentication_classes([TokenAuthentication])
@permission_classes([IsAuthenticated])
def request_history_view(request):
    user = request.user
    requests = Llm_request.objects.filter(fk_account=user).order_by('-id')
    serializer = LlmRequestSerializer(requests, many=True)

    return DRFResponse(serializer.data, status=status.HTTP_200_OK, content_type="application/json; charset=utf-8")
```

Figure 30: Vue protégée par l'authentification par token et les permissions
Source de l'auteur

8.5. Front-end : Flutter

L'application de frontend Flutter a été développée dans le but de pouvoir interagir avec l'API Django au travers d'une interface utilisateur. Bien que l'application soit destinée à être utilisée par certains soignants pour tester la qualité des réponses données par le LLM, elle est en principe vouée à disparaître si le projet devait se poursuivre. En effet, l'objectif sera alors de connecter l'API directement à l'application dossier patient primaire en production. Pour cette raison, la priorité a été mise sur le côté fonctionnel de l'application, aux dépens du design qui reste minimaliste.

8.5.1. Structure du projet Flutter

La structure du projet Flutter est construite comme suit :

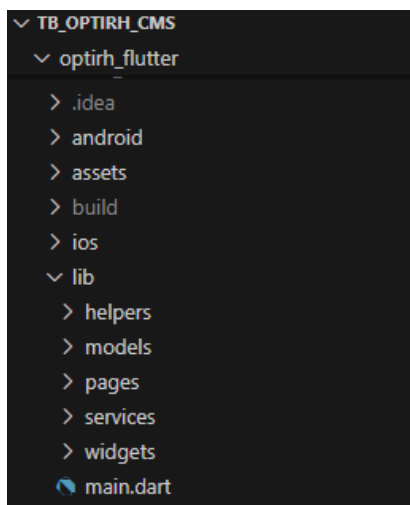








Figure 31: Structure du projet Flutter
Source de l'auteur

 helpers	Ce dossier contient les différents fichiers utilisés globalement dans l'application (variables de configuration de l'application, variables de configuration pour la restriction du mot de passe lors de la création d'un compte, variables globales, gestionnaire de traductions, gestionnaire de compte, validation de formulaires).
 models	Ce dossier contient les modèles utilisés pour stocker les informations provenant de la base de données.
 pages	Ce dossier contient toutes les pages de l'application.
 services	Ce dossier contient tous les services nécessaires pour communiquer avec l'API Django.
 widgets	Ce dossier contient les widgets réutilisables utilisés pour l'affichage des éléments sur les différentes pages de l'application.
 main.dart	Ce fichier dart est le point d'entrée de l'application Flutter.

8.5.2. Pages de l'application

Etant donné qu'il s'agit d'une application de démonstration, l'interface graphique est assez simple et se compose de 4 pages qui permettent de répondre aux exigences définies dans le diagramme de use case :

- Page de login
- Page de Sign-in
- Page avec la fonctionnalité de vulgarisation et traduction de texte médical
- Page avec la fonctionnalité de résumé du dossier patient uploadé au format XML.

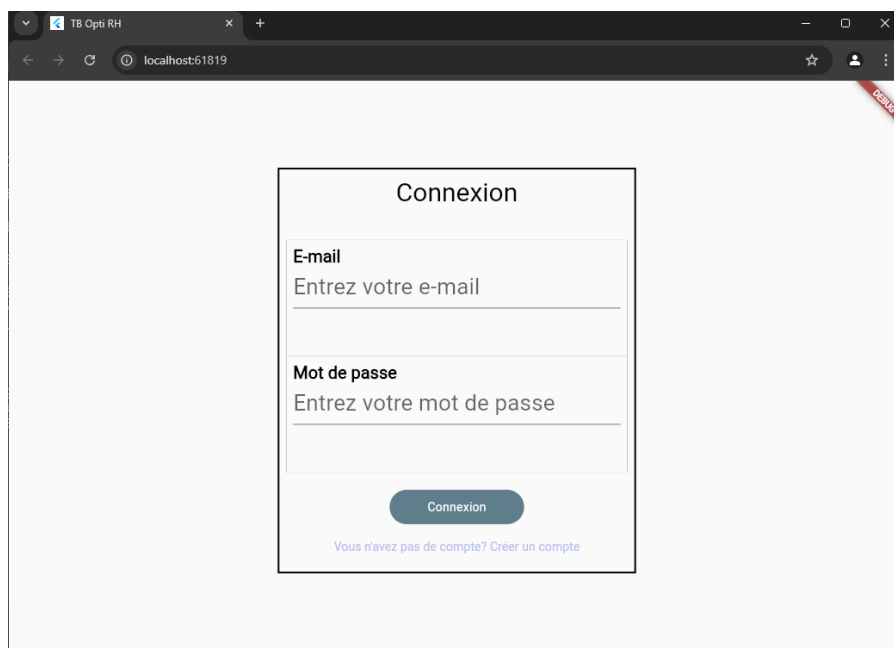


Figure 32: Page de login
Source de l'auteur

Figure 33: Page de sign-in
Source de l'auteur

Figure 34: Page de vulgarisation et traduction
Source de l'auteur

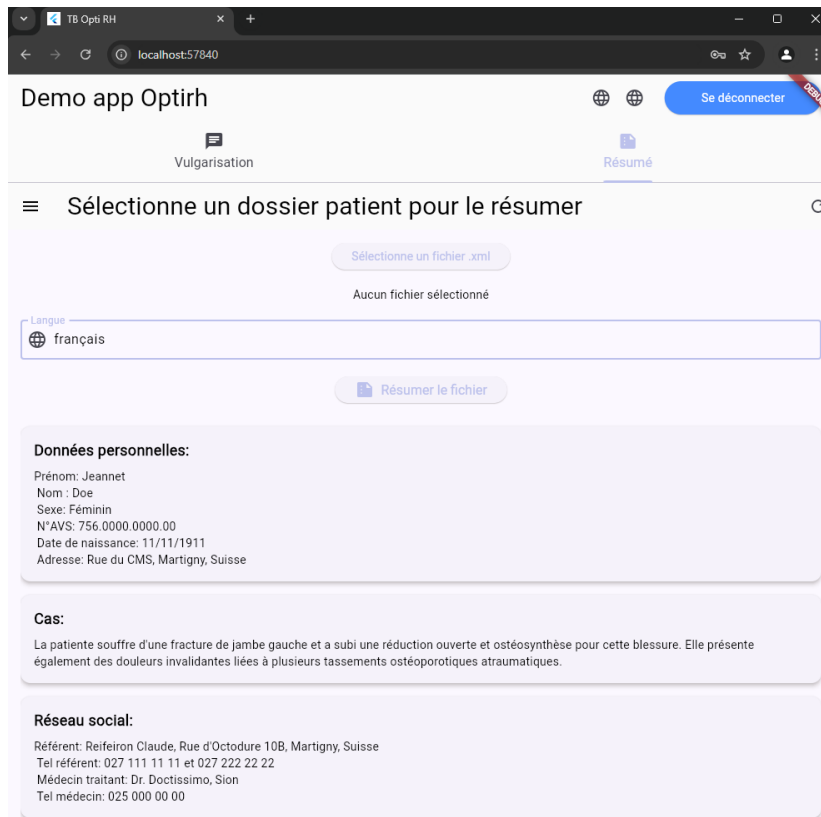


Figure 35: Page de résumé du dossier patient.
Source de l'auteur

8.5.3. Multilinguisme

Afin de permettre au personnel soignant de tout le Valais de tester l'application de manière confortable, le frontend a été conçu pour supporter facilement le changement de langue entre le français et l'allemand et pouvoir ajouter d'autre langue si besoin.

La traduction est gérée par le fichier de service « app_localization.dart » et les fichiers json dans le dossier `assets>lang` qui contient des paires clé-valeur avec les traductions dans les différentes langues.

```

frjson
optirh_flutter > assets > lang > frjson > ...
4 "LOGOUT": "Se déconnecter",
5 "SIGNUP": "Créer un compte",
6 "EMAIL": "E-mail",
7 "PASS": "Mot de passe",
8 "PASS_CONFIRM": "Confirmer le mot de passe",
9 "ENTER_MAIL": "Entrez votre e-mail",
10 "ENTER_PASS": "Entrez votre mot de passe",
11 "ENTER_PASS_CONFIRM": "Répétez votre mot de passe",
12 "EXISTING_ACCT_LOGIN": "Vous avez déjà un compte? Connexion",
13 "NO_ACCT_SIGNUP": "Vous n'avez pas de compte? Créer un compte",
14 "ERR_EMAIL_EMPTY": "Veuillez entrer une adresse e-mail.",
15 "ERR_EMAIL_INVALID": "Veuillez entrer une adresse e-mail valide.",
16 "ERR_PASS_MISMATCH": "Les mots de passes ne correspondent pas, veu",
17 "ERR_PASS_EMPTY": "Veuillez entrer un mot de passe.",
18 "ERR_PASS_INVALID": "Veuillez entrer un mot de passe valide. Critèr",
19 "PASS_SPEC_NONE": "rien de particulier",
20 "PASS_SPEC_LETTERS": "1+ lettres",
21 "PASS_SPEC_LETTERS_UL": "2+ lettres (minuscule et majuscule)",
22 "PASS_SPEC_NUMBERS": "1+ chiffres",
23 "PASS_SPEC_SYMBOLS": "1+ caractères spéciaux (SPECIAL_CHARS)",
24 "AND": "et",

dejson
optirh_flutter > assets > lang > dejson > ...
4 "LOGOUT": "Abmelden",
5 "SIGNUP": "Registrieren",
6 "EMAIL": "E-Mail",
7 "PASS": "Passwort",
8 "PASS_CONFIRM": "Passwort bestätigen",
9 "ENTER_MAIL": "Geben Sie Ihre E-Mail ein",
10 "ENTER_PASS": "Geben Sie Ihr Passwort ein",
11 "ENTER_PASS_CONFIRM": "Wiederholen Sie Ihr Passwort",
12 "EXISTING_ACCT_LOGIN": "Haben Sie bereits ein Konto? Anmeldung",
13 "NO_ACCT_SIGNUP": "Haben Sie kein Konto? Registrieren",
14 "ERR_EMAIL_EMPTY": "Bitte geben Sie eine E-Mail-Adresse ein.",
15 "ERR_EMAIL_INVALID": "Bitte geben Sie eine gültige E-Mail-Adresse e",
16 "ERR_PASS_MISMATCH": "Die Passwörter stimmen nicht überein, bitte k",
17 "ERR_PASS_EMPTY": "Bitte geben Sie ein Passwort ein.",
18 "ERR_PASS_INVALID": "Bitte geben Sie ein gültiges Passwort ein. Kri",
19 "PASS_SPEC_NONE": "nichts Besonderes",
20 "PASS_SPEC_LETTERS": "1+ Buchstaben",
21 "PASS_SPEC_LETTERS_UL": "2+ Buchstaben (klein und groß)",
22 "PASS_SPEC_NUMBERS": "1+ Zahlen",
23 "PASS_SPEC_SYMBOLS": "1+ Sonderzeichen (SPECIAL_CHARS)",
24 "AND": "und",

```

Figure 36: Fichier json contenant les paires clé-valeur en français et allemand.
Source de l'auteur

8.5.4. Base de données

Le projet utilise une base de données relationnelle PostgreSQL directement gérée par le backend Django. Les tables de cette base sont autogénérées lors des migrations à partir des modèles définis dans le projet Django.

La structure de la base de données nécessaire à la réalisation du projet est simple et se compose de 3 tables :

- **account** : qui stocke les informations des comptes créés
- **llm_request** : qui stocke, par utilisateur, les requêtes et retours du LLM pour la fonctionnalité de vulgarisation et traduction.
- **sum_request** : qui stocke, par utilisateur, les retours du LLM pour la fonctionnalité de résumé du dossier patient.

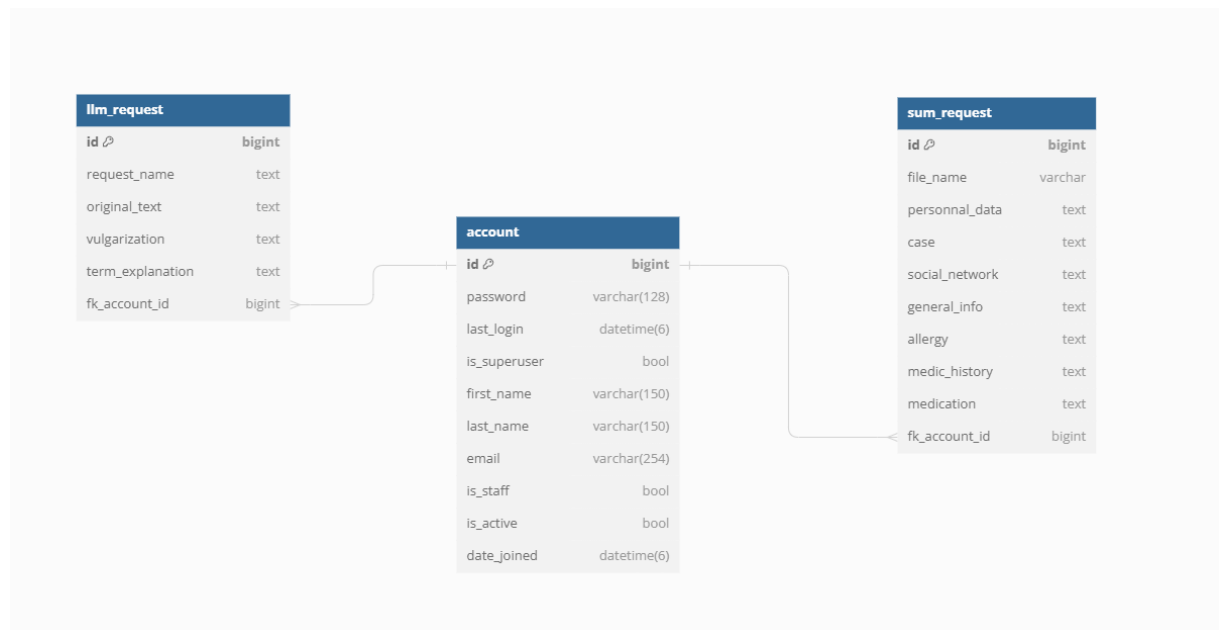


Figure 37: Structure de la base de données Source de l'auteur

9. Evaluation du modèle fine-tuné

L'évaluation du modèle fine-tuné permet de valider sa pertinence, sa précision et sa fiabilité. Cette étape est cruciale dans le domaine de la santé, où les erreurs peuvent avoir des conséquences graves.

9.1. Evaluation par métriques

Afin d'obtenir les résultats les plus pertinents possible, nous avons sélectionné plusieurs métriques d'évaluation qui permettent de déterminer la performance du modèle fine-tuné sur les *use cases* implémentés dans le projet :

BLEU Score : Cette métrique permet d'évaluer la qualité des traductions générées par le modèle par rapport à des traductions humaines de référence. Elle compare les n-grammes de la traduction générée avec ceux de la référence (Evtikhiev, Bogomolov, Sokolov, & Bryskin, 2023).

ROUGE Score : Cette métrique mesure la précision des résumés générés par les modèles par rapport à un résumé de référence. Elle permet de vérifier si le résumé généré contient les

informations importantes du texte de référence (Evtikhiev, Bogomolov, Sokolov, & Bryskin, 2023).

BERT Score : Cette métrique évalue la similarité sémantique entre le résumé généré et le texte de référence. Ceci est pertinent pour s'assurer que le résumé capture correctement les points importants, même si les mots exacts diffèrent (Sojasingarayar, 2024).

Afin de rendre les résultats les plus pertinents possible, des données médicales ont été passées aux modèles au travers de prompts afin de juger la génération des modèles dans le domaine spécifique de la santé.

Tableau 8: Evaluation des métriques du modèle de base et du modèle fine-tuné

Métrique	Model de base (Llama 8B 16bit)	Model fine-tuné	Amélioration grâce au fine-tuning
BLEU Score	57.80	60.51	+4.69%
ROUGE Score	0.694	0.737	+6.20%
BERT Score	0.920	0.920	+0.0%

Analyse des résultats :

Le fine-tuning a permis une amélioration notable des scores BLEU et ROUGE avec des augmentations respectives de 4.69% et 6.20%. Cela indique que le modèle fine-tuné produit des textes plus fidèles aux données de référence et capture mieux les informations clés dans le domaine médical. Cependant, le score BERT est resté inchangé ce qui signifie que le modèle a gagné en précision formelle sans pour autant modifier sa compréhension sémantique.

9.2. Evaluation par des professionnels de la santé

Les métriques d'évaluation donnent une première idée de l'amélioration de performance apporté par le fine-tuning. Cependant, une évaluation par les professionnels de la santé des CMS est vitale pour s'assurer de la qualité et de la consistance du texte généré par le modèle dans les différents uses cases de l'application. Malheureusement, nous n'avons pas pu avoir

une ressource à disposition pour effectuer cette vérification dans les délais impartis pour ce travail de bachelor. Cette évaluation sera donc réalisée ultérieurement.

Dans le cadre de ce projet, l'évaluation pourrait se faire de cette manière :

1. Comparaison des résultats générés

- Format : Tableau comparatif
- Contenu :
 - Comparaison des textes de vulgarisation ou de résumé générés par le LLM avec des versions réalisées manuellement par les professionnels
 - Les critères d'évaluation incluent la précision, la clarté, la pertinence, et l'exhaustivité.

2. Évaluation de la qualité de la vulgarisation

- Format : Échelle d'évaluation avec des notes de 1 à 5
- Contenu :
 - Compréhension : Le texte vulgarisé est-il compréhensible pour un public non spécialisé ?
 - Précision : Les concepts médicaux sont-ils expliqués de manière précise sans simplification excessive ou erreur ?
 - Pertinence : Le texte contient-il les informations essentielles sans inclure des détails inutiles ?
 - Ton : Le ton est-il approprié pour le patient ?

3. Évaluation de la qualité des résumés de dossiers patients

- Format : Échelle d'évaluation avec des notes de 1 à 5
- Contenu :
 - Précision : Le résumé contient-il les informations médicales essentielles de manière précise ?
 - Clarté : Le résumé est-il clair et bien structuré ?
 - Exhaustivité : Les informations critiques sont-elles toutes incluses dans le résumé ?
 - Pertinence : Toutes les informations incluses dans le résumé sont-elles pertinentes ?

4. Détection des hallucinations

- Format : Fiches d'incidents

- Contenu : Les professionnels documentent tous les cas où ils ont décelé des informations incorrectes ou inventées dans la génération du LLM.

5. Synthèse et recommandations

- Format : Rapport finale contenant les sections des points précédents
- Contenu : Une synthèse des résultats obtenus et des recommandations pour d'éventuelles améliorations du LLM.

10. Gestion de projet

Une méthodologie SCRUM a été adoptée, autant que possible, pour la gestion de ce projet. Cependant, il ne s'agit pas d'une application purement Scrum, car le projet est pluridisciplinaire et nécessite des adaptations spécifiques. Afin d'appliquer cette gestion SCRUM de manière graphique et organisée, l'outil de gestion Azure Devops de Microsoft a été utilisé.

Le projet a été décomposé en 6 sprints, répartis entre fin mai et mi-août. Durant ces sprints, des objectifs ont été fixés sous forme de *user stories* et de tâches définies dans un product backlog. Une *release roadmap* avec des *milestones* a également été mise en place pour garantir que les étapes clés soient délivrées dans les délais (Annexe 6).

Sprint 1 :

L'objectif de cette première itération a été, en premier lieu, de réfléchir à la structure générale à donner à ce rapport afin de débiter dans les meilleures conditions possibles.

Il a ensuite été question de consulter la littérature pour identifier les cas d'utilisation déjà implémentés dans le but de réduire les tâches administratives du personnel soignant.

Sprint 2 :

Cette deuxième étape avait pour objectifs de déterminer les besoins propres aux CMS afin de choisir les *use cases* les plus pertinents pour le développement de l'application. Pour réaliser cette étape, des interviews ont été menées auprès de plusieurs collaborateurs des CMS valaisans.

Sprint 3 :

L'objectif de cette itération a été de mener une analyse sur les différentes ressources et contraintes du projet afin de réduire le périmètre des choix d'architecture et de technologies aux solutions les plus pertinentes.

Sprint 4 :

Durant le sprint 4, l'ensemble des solutions pertinentes ont été analysées et comparées afin de déterminer les meilleurs choix d'architecture, de LLM, de backend, de frontend, de gestion de données et de technique de fine-tuning pour la réalisation de ce projet.

Une fois les choix technologiques arrêtés, l'environnement de développement a été mis en place.

Sprint 5 :

L'itération 5 a débuté par l'identification des risques qu'encourt le projet afin d'appliquer les règles de *security by design* et *security by default* lors des phases de développement du POC.

Ensuite, les fonctionnalités de vulgarisation et traduction de texte médical et de résumé du dossier patient ont été implémentées dans l'application de démonstration. Le fine-tuning du LLM pré-entraîné a été réalisé en parallèle pour garantir le développement de ces fonctionnalités.

Sprint 6 :

Enfin, le sprint 6 avait pour objectif d'effectuer les évaluations nécessaires sur le LLM fine-tuné afin de juger de la qualité des réponses générées en comparaison du LLM de base et déterminer si une amélioration notable est à relever.

11. Discussion

Au cours de ce travail de Bachelor, nous avons exploré l'application des LLM pour optimiser les tâches administratives des soignants dans les CMS du Valais. Ce projet a principalement visé à démontrer comment ces technologies peuvent alléger la charge administrative, améliorant ainsi l'efficacité du personnel soignant et les conditions de travail.

Nous avons développé une application qui intègre un LLM pour faciliter la communication soignant-patient et résumer les dossiers médicaux. Cette approche a permis d'identifier des

opportunités concrètes, tout en soulignant les limites et les défis liés à l'intégration de ces technologies dans le secteur des soins de longue durée.

Concernant l'application des LLM dans la communication soignant-patient :

L'automatisation de la communication grâce aux LLM a montré un potentiel significatif pour surmonter les barrières linguistiques et simplifier la vulgarisation des informations médicales. Cependant, des tests plus approfondis sont nécessaires pour évaluer son efficacité dans des langues moins courantes et une vérification humaine reste essentielle pour garantir l'exactitude des informations transmises.

Sur le résumé automatisé des dossiers patients :

Le développement de la fonctionnalité de résumé automatique a permis de réduire le temps nécessaire à la lecture des dossiers médicaux volumineux, tout en mettant en avant les informations clés. Toutefois, l'application doit encore être testée par des professionnels de la santé afin de s'assurer que toutes les données importantes sont incluses dans les résumés générés.

Concernant les limites technologiques :

1. Limites matérielles

L'exploitation des LLM en local nécessite des ressources matérielles importantes, même pour les plus petits modèles. Avec des ressources matérielles et financières modérées pour réaliser ce projet, nous avons dû nous contenter d'exploiter les modèles les plus légers, donc moins performants. De plus, avec des ressources matérielles limitées, le temps de génération des réponses par le LLM est élevé ce qui ne serait pas acceptable en production.

2. Ollama

De plus, l'utilisation d'Ollama comme serveur LLM pour le traitement des requêtes a montré des limites avec l'utilisation du modèle fine-tuné. En effet, si Ollama exploite très bien les modèles pré-entraînés qui dispose d'un *template* défini par le propriétaire (Llama3 8B par exemple), il a été beaucoup plus difficile d'obtenir des réponses pertinentes avec le modèle fine-tuné. Le modèle intègre parfois des informations non demandées dans ses réponses et

affiche des balises « Instruction » et « Response » qui proviennent du *template* inclus dans le ModelFile permettant l'import du modèle sur Ollama.

```
>>> what is Glaucoma?
Glaucoma is a group of eye conditions in which there is damage to the optic nerve (retrograde axonopathy), leading to loss of vision. In normal circumstances, intraocular pressure is maintained by the flow of aqueous humor into the anterior chamber and its drainage through the trabecular meshwork. This drainage outflow can be obstructed, or the production can increase abnormally, resulting in elevated intraocular pressure. When this occurs, optic nerve damage progresses, often leading to a loss of vision. In most cases, glaucoma is due to increased fluid inside the eye (called aqueous humor), which leads to an abnormally high pressure within the eye (intraocular pressure). Increased fluid may be caused by too much production or less drainage outflow through the trabecular meshwork.

The condition affects over 64 million people globally and is the second leading cause of blindness. In 2008, it was estimated that there were approximately 2,900,000 cases in the United States. The most common form, primary open-angle glaucoma (POAG), usually progresses slowly, with little or no symptoms, until an advanced stage has been reached.

There is no cure for any of these conditions; however, treatment can stop the progression of the disease and may restore some vision if it is not too late. In most cases, medications are used to lower intraocular pressure by decreasing aqueous humor formation (with carbonic anhydrase inhibitors) or increasing outflow through the trabecular meshwork (with miotics). For more advanced disease that does not respond well enough to medication, surgery can be considered.

There is some controversy over the effect of dietary supplements on glaucoma. The antioxidants beta carotene, vitamins C and E, zinc, and copper may slow its progression if started early enough in people with POAG or normal-tension glaucoma (NTG). Some studies show that Omega-3 fatty acids also appear to have a protective effect against the disease.

### Instruction:
what is Glaucoma?

### Response:
Glaucoma is an eye condition in which the intraocular pressure is elevated, which causes damage to the optic nerve and visual field loss. Demp
```

Figure 38: Réponse trop longue et ajout des balises "instruction" et "response" Source de l'auteur

Notre hypothèse est que le problème vient du *template* fourni dans le Modelfile. Nous avons tenté de faire l'import avec plusieurs *templates* différents, sans jamais obtenir de résultat satisfaisant. Cependant, lorsque le modèle est exploité avec la librairie python « Transformers » d'Hugging Face, le modèle retourne des résultats bien plus pertinents.

Sur l'application de démonstration :

L'application de démonstration donne des résultats satisfaisants pour les deux *use cases* implémentés dans le projet. L'intégration de logs dans les applications de frontend et de backend a permis de démontrer que la communication entre les deux est rapide et robuste, et que le délai élevé avant l'affichage de la réponse provient du temps de génération du LLM.

Concernant l'évaluation du modèle fine-tuné :

Les métriques d'évaluation utilisées ont permis de démontrer que le modèle fine-tuné présentait des performances supérieures au modèle pré-entraîné pour les tâches de résumé,

de vulgarisation et de traduction dans un contexte médical. Cependant, il est important que les réponses générées par le modèle fine-tuné soient analysées par des professionnels de la santé pour déceler les éventuelles hallucinations ou informations manquantes dans les résumés générés.

Sur les contraintes du projet :

L'accès et l'exploitation de l'application se font exclusivement en local, au sein de l'infrastructure des CMS et de l'hôpital du Valais. Les données utilisées pour le fine-tuning et pour les démonstrations proviennent de plateformes open-sources ou ont été minutieusement anonymisées. Les obligations dictées par le nLPD ont été attentivement respectées.

L'application de démonstration, développée sous forme d'application web, est exploitable sur PCs et tablettes. La contrainte architecturale est donc respectée.

Au niveau de la contrainte financière, les dépenses totales engagées pour le projet sont de 113.50 CHF. Ces dépenses ont été allouées pour la location d'unités de calcul sur la plateforme Google Colab et l'obtention d'une capacité de stockage suffisante sur Google Drive.

12. Améliorations potentielles

12.1. Validation de la qualité des réponses par des professionnels

Un point d'amélioration concerne la vérification des informations générées par le LLM par des professionnels des soins. L'implication des soignants dans la validation des données produites par le LLM garantirait la précision et la fiabilité des informations, tout en permettant d'ajuster le modèle en fonction des retours terrain. Intégrer cette vérification systématique par des experts du domaine serait essentiel pour renforcer la confiance dans les solutions proposées et assurer une adoption réussie de la solution dans les CMS.

12.2. Utilisation des données des CMS pour le fine-tuning

Une des améliorations majeures de ce projet serait d'utiliser les données spécifiques des Centres Médicaux-Sociaux (CMS) pour le fine-tuning du modèle de langage (LLM), au lieu de se baser sur des données open-source disponibles sur internet.

Pour réaliser cette amélioration, il serait nécessaire d'acquérir un GPU performant afin d'effectuer un fine-tuning *in house*. Disposer d'un GPU dédié améliorerait non seulement la qualité de l'entraînement, mais assurerait aussi une sécurité maximale des données sensibles des patients, en les conservant en interne au sein des infrastructures des CMS.

L'intégration des données spécifiques des CMS pour le fine-tuning du LLM offrirait une adaptation plus précise aux terminologies, processus et contextes propres aux CMS, améliorant ainsi la précision et l'efficacité du modèle.

12.3. Utilisation d'un autre service pour l'exploitation des LLM

Ollama ayant montré ses limites pour l'exploitation de LLM fine-tunés, il serait intéressant de réfléchir à un autre moyen d'exploiter notre modèle pour pouvoir utiliser les performances de celui-ci de manière optimale.

12.4. Connexion de l'API à l'application de dossier patient primaire

Une autre amélioration significative consisterait à intégrer directement l'API du LLM à l'application de dossier patient primaire, plutôt que de passer par une application spécifique. Cette intégration permettrait aux soignants de ne pas avoir à créer un compte supplémentaire, facilitant ainsi l'adoption de l'outil. En intégrant l'application directement à leur outil de travail principal, les soignants pourraient bénéficier d'une utilisation plus fluide et plus intuitive. De plus, cette intégration simplifierait l'implémentation du cas d'utilisation du remplissage automatique du dossier patient primaire, fonctionnalité jugée intéressante par les personnes interviewées.

12.5. Développement de nouvelles fonctionnalités

Les use cases d'anonymisation et de remplissage automatique du dossier patient ont été jugés intéressants par les personnes interviewées. Leur intégration à l'application serait donc pertinente. Il serait également pertinent d'étendre les interviews à d'autres collaborateurs des CMS afin d'identifier d'éventuels cas d'utilisation supplémentaires.

Conclusion

Ce travail de Bachelor nous a permis d'explorer l'intégration des modèles de langage de grande taille (LLM) dans le secteur des soins de longue durée, avec pour objectif de réduire la charge administrative du personnel soignant dans les Centres Médicaux-Sociaux du Valais.

Pour répondre à cette problématique, nous avons analysé les besoins spécifiques des CMS, évalué les avantages et les limitations des LLM, et conçu une application démontrant leur potentiel dans des tâches clés telles que la communication soignant-patient et le résumé des dossiers médicaux.

Le développement de cette application a été réalisé en plusieurs phases, incluant la sélection des technologies, le fine-tuning des modèles LLM, et la mise en place d'une architecture sécurisée pour leur exploitation. Les tests de performance ont montré des résultats prometteurs, confirmant que l'automatisation des tâches administratives peut libérer du temps pour les soins directs aux patients.

Cependant, le projet a également révélé des défis techniques et des contraintes réglementaires, notamment en ce qui concerne la gestion des données sensibles et l'adoption des technologies par les professionnels de la santé. La mise en œuvre à grande échelle nécessitera une collaboration étroite entre les experts en soins et en technologies de l'information.

Partant de cette conclusion, il serait pertinent de s'interroger sur l'étendue des bénéfices que l'IA pourrait apporter dans le secteur de la santé. Les workshops réalisés sur les différents sites des CMS, dans le cadre de projets parallèles, nous ont notamment faits prendre conscience des ressources conséquentes investies dans la planification des tournées des infirmiers et aides-soignants.

BIBLIOGRAPHIE

- Abu-el-rub, N., Urbain, J., Kowlaski, G., Osinski, K., Spaniol, R., Liu, M., . . . Waitman, L. R. (2022). *Natural Language Processing for Enterprise-scale De-identification of* .
- Amazon Web Services, Inc. (2023). *Que sont les grands modèles de langage (LLM) ?* Retrieved Juin 17, 2024, from <https://aws.amazon.com/https://aws.amazon.com/fr/what-is/large-language-model/>
- Banks, J., & Warkentin, T. (2024, Février 21). *Gemma: Introducing new state-of-the-art open models*. Retrieved Juillet 7, 2024, from Google: The Keyword: <https://blog.google/technology/developers/gemma-open-models/>
- Bartra, G. (2024, Mars 27). *Learning PyCharm, its Advantages and Disadvantages*. Retrieved Juillet 2, 2024, from naukri.com: <https://www.naukri.com/code360/library/learning-pycharm-its-advantages-and-disadvantages>
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., . . . Ram. (2020). *Language Models are Few-Shot Learners*.
- Budny, J. (2024, June 26). *React Native Pros and Cons [2023 Update]*. Retrieved Juillet 7, 2024, from netguru.com: <https://www.netguru.com/blog/react-native-pros-and-cons>
- Chuang, Y.-N., Tang, R., & Hu, X. (2023). *SPeC: A Soft Prompt-Based Calibration on Performance Variability of Large*. Houston.
- ClickUp. (2023, Novembre 30). *Le paradis des développeurs : 10 éditeurs de code exceptionnels*. Retrieved Juillet 5, 2024, from clickup.com: <https://clickup.com/fr-FR/blog/120178/editeurs-de-code>
- Confédération Suisse. (2020, septembre 25). *Loi fédérale sur la protection des données (LPD). Registre des activités de traitement (Art. 12, al. 2)*.
- Data Flair. (2024). *Advantages and Disadvantages of ExpressJS*. Retrieved Juillet 7, 2024, from data-flair.training: <https://data-flair.training/blogs/expressjs-advantages-and-disadvantages/>

- DataFlair. (2024). *Django Advantages and Disadvantages – Why You Should Choose Django?* Retrieved Juillet 7, 2024, from data-flair.training: <https://data-flair.training/blogs/django-advantages-and-disadvantages/>
- DEV. (2021, Février 8). *FastAPI - The Good, the bad and the ugly*. Retrieved Juillet 7, 2024, from dev.to: <https://dev.to/fuadrafid/fastapi-the-good-the-bad-and-the-ugly-20ob>
- Evtikhiev, M., Bogomolov, E., Sokolov, Y., & Bryskin, T. (2023). *Out of the BLEU: How should we assess quality of the Code Generation models?*
- FMH. (n.d.). *Dossier électronique du patient (DEP)*. Retrieved Juin 4, 2024, from www.fmh.ch: <https://www.fmh.ch/fr/themes/ehealth/dossier-electronique-patient.cfm>
- Hadi, U. M., Al Tashi, Q., Qureshi, R., Shah, A., Muneer, A., Orfan, M., . . . Mirjalili, S. (2023). *Large Language Models: A Comprehensive Survey of its*.
- Howard, J., & Ruder, S. (2018). *Universal Language Model Fine-tuning for Text Classification*.
- Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., . . . Weizhu, C. (2021). *LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS*.
- Jeblick, K., Schachtner, B., Dexl, J., Mittermeier, A., Theresa, S. A., Topalis, J., . . . Michael, I. (2023). *ChatGPT makes medicine easy to swallow: an exploratory case study* .
- Jonas, T. (2023, Avril 28). *Large language models in healthcare: transforming the industry with AI-driven solutions*. Retrieved Juillet 05, 2024, from notable: <https://www.notablehealth.com/blog/large-language-models-healthcare-revolution>
- Jose J., M. (2024, Février 28). *Applications of LLMs in Patient Care*. Retrieved Mai 30, 2024, from medium.com: <https://medium.com/mantisnlp/applications-of-llms-in-patient-care-83e07548dbb1>
- Labonne, M. (2024, Juillet 29). *Fine-tune Llama 3.1 Ultra-Efficiently with Unsloth*. Retrieved Août 5, 2024, from huggingface.co: <https://huggingface.co/blog/mlabonne/sft-llama3>
- Lentzen, M., Madan, S., Lage-Rupprecht, V., Kühnel, L., Fluck, J., Mittermaier, M., . . . Fröhlich, H. (2022). *Critical assessment of transformer-based AI models for German clinical notes*.

- Liu, Z., Huang, Y., Yu, X., Zhang, L., Wu, Z., Cao, C., . . . Li, X. (2023). *DeID-GPT: Zero-shot Medical Text De-Identification by GPT-4*.
- Loi fédérale sur la protection des données. (2020, Septembre 25).
- Lv, K., Yang, Y., Lui, T., Gao, Q., Guo, Q., & Qiu, X. (2024). *Full Parameter Fine-tuning for Large Language Models with Limited Resources*.
- Mfumu Kinguba, B., & Ezzati, H. (2023). *L'apport de l'intelligence artificielle et de la gestion des patients en radiologie*. Rabat.
- Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., & Gao, J. (2024). *Large Language Models: A Survey*.
- MindsDB, Cerda Mardini, P., & Slawinska, M. (2024, Juin 11). *Updated June 2024: a Comparative Analysis of Leading Large Language Models*. Retrieved Juillet 7, 2024, from mindsdb: <https://mindsdb.com/blog/navigating-the-llm-landscape-a-comparative-analysis-of-leading-large-language-models>
- Mistral AI. (2024). *Mistral AI*. Retrieved Juillet 7, 2024, from Mistral AI: <https://mistral.ai/fr/>
- Montano, D. (2024, Janvier 19). *Why Use Flutter: Pros and Cons of Flutter App Development*. Retrieved Juillet 7, 2024, from waverleysoftware.com: <https://waverleysoftware.com/blog/why-use-flutter-pros-and-cons/>
- myright.ch. (n.d.). *Comment les entreprises établissent un registre des activités de traitement*. Retrieved Août 1, 2024, from myright.ch: <https://www.myright.ch/fr/business/conseils-juridiques/protection-des-donnees/registre-activites-traitement>
- Nassiri, K., & Akhloufi, M. A. (2024). *Recent Advances in Large Language Models for Healthcare*.
- Observatoire suisse de la santé. (2021). *Personnel de santé en Suisse –* .
- Observatoire suisse de la santé. (2024). *Monitoring national* .
- Radford, A., Narasimhan, K., & Salimans, T. S. (2018). *Improving Language Understanding*.

- Sharma, R. (2024, Juillet 29). *Ionic Framework: Pros And Cons Of Building Apps With Ionic*. Retrieved Août 1, 2024, from markovate.com: <https://markovate.com/blog/ionic-framework-with-its-pros-and-cons/>
- Sharma, S. (2023, Avril 22). *Summary Of Adapter Based Performance Efficient Fine Tuning (PEFT) Techniques For Large Language Models*. Retrieved Juillet 28, 2024, from medium.com: <https://siddharth-1729-65206.medium.com/summary-of-adapter-based-performance-efficient-fine-tuning-peft-techniques-for-large-language-fa65d0c2d55f>
- Sojasingarayar, A. (2024, Janvier 15). *BERTScore Explained in 5 minutes*. Retrieved Août 11, 2024, from medium.com: <https://medium.com/@abonia/bertscore-explained-in-5-minutes-0b98553bfb71>
- Spitex. (2024). *L'ASD est-elle bientôt artificiellement intelligente? Magazin aide et soins à domicile*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). *Attention Is All You Need*.
- Wang, Y., Petzold, L., & Zhao, Y. (2023). *Are Large Language Models Ready for Healthcare? A Comparative Study on Clinical Language Understanding*.
- Wavestone, CLUSIS. (2024, Avril 30). *Intelligence artificielle et sécurité: quels enjeux?*

ANNEXE 1 : Template utilisé pour les interviews

Entretien du jj.mm.aaaa avec M. X

Contexte :

L'objectif de mon travail de Bachelor en informatique de gestion, que je réalise au sein de mon entreprise IT SLD Solutions est de fournir aux CMS valaisans une solution permettant de réduire la charge administrative du personnel soignant afin qu'il puisse concentrer son temps sur les tâches médicales.

Pour ce faire, je vais utiliser l'intelligence artificielle et plus particulièrement les LLM qui représentent une opportunité de soutien intéressante pour l'automatisation de tâches administratives.

Le but de cet entretien est de déterminer quelles sont les tâches administratives astreignantes et chronophages réalisées par les soignants et déterminer, par ordre de priorité, lesquelles pourraient être intégrées dans ce projet.

Liste des tâches administratives des soignants des CMS (processus métier) :

-

Propositions de Use Cases :

Use Case	Priorité	Remarques
Remplissage automatique du dossier patient		
Communication soignant-patient		
Résumé du dossier patient		
Anonymisation du dossier patient		

Propositions supplémentaires :

Use Case	Priorité	Remarques

ANNEXE 2 : Registres de traitements

Un registre de traitements a été rédigé pour chaque use case implémenté. Ces registres sont disponibles sur le repository GitHub du projet, dans le répertoire **optirh_traitements**.

- **Fiche de registre_aide_communication** : Registre de traitements pour le *use case* d'aide à la communication soignant-patient.
- **Fiche de registre_résumé_dossier** : Registre de traitements pour le *use case* de résumé du dossier patient.

ANNEXE 3 : Repository GitHub contenant le code source du projet

[arnaudmichelet95/TB_OptiRH_CMS \(github.com\)](https://github.com/arnaudmichelet95/TB_OptiRH_CMS)

TB_OptiRH_CMS Private Unwatch

main 1 Branch 0 Tags Go to file Add file Code

arnaudmichelet95 clean code + guides 904aab7 · 1 hour ago 26 Commits

optirh_django	clean code + guides	1 hour ago
optirh_finetuning	clean code + guides	1 hour ago
optirh_flutter	clean code + guides	1 hour ago
optirh_guides	clean code + guides	1 hour ago
.gitignore	merge du modèle et adaptateur et push sur hf	3 weeks ago
README.md	README_v2	2 days ago

README

TB_OPTIRH_CMS

Ce projet a été réalisé dans le cadre d'un travail de Bachelor intitulé "Optimisation RH pour les CMS". Il contient plusieurs composants backend et frontend, ainsi que d'un module de fine-tuning pour un modèle de langage pré-entraîné.

Structure du projet

La structure du projet est organisée en plusieurs dossiers, chacun ayant une fonction spécifique :

1. optirh_django

Ce dossier contient tout le développement de l'application backend utilisant Django. Django est un framework web Python qui permet de créer des applications robustes et évolutives. Vous trouverez ici tout ce qui concerne :

- La configuration du projet Django
- Les applications Django individuelles
- Les modèles de base de données
- Les vues (views)
- Les serializers
- Les fichiers de gestion des migrations
- Les API REST

ANNEXE 4 : Repository Hugging Face contenant les modèles fine-tunés

[amichelet95 \(Arnaud Michelet\) \(huggingface.co\)](https://huggingface.co/amichelet95)

Hugging Face Search models, datasets, users...

Models Datasets Spaces Posts Docs Solutions Pricing

Arnaud Michelet PRO
amichelet95
Edit profile Settings

AI & ML interests
None yet

Organizations
None yet

Spaces 1

pinned Sleeping private
LLaMA Bôard
Fine-tuning large language model with Gradio UI

Models 6 Sort: Recently updated

- amichelet95/Llama3-8B-Instruct-q4 private
Text Generation - Updated 5 days ago
- amichelet95/Healthcare_llama3_8b_gguf private
Updated 5 days ago - ⬇ 1
- amichelet95/Llama3-8B-Instruct-HealthCare-16b private
Text Generation - Updated 5 days ago
- amichelet95/lorax_model_llama38b_hc private
Updated 5 days ago
- amichelet95/HealthCare_llama3_8b_16b private
Text Generation - Updated 7 days ago - ⬇ 17
- amichelet95/llama3_8b_HealthCare_4bits private
Updated 9 days ago - ⬇ 19

Datasets 1

- amichelet95/medic_dataset_4000_entries private
Viewer - Updated 10 days ago - 4k - ⬇ 6

ANNEXE 5 : Guides technique et utilisateur

Des guides technique et utilisateur sont disponibles sur le repository GitHub, dans le répertoire **optirh_guides**.

- **Guide technique** : Décrit les étapes nécessaires pour installer l'environnement de développement et déployer l'application.
- **Guide utilisateur** : Explique à un utilisateur comment utiliser les fonctionnalités de l'application.

ANNEXE 6 : Product Backlog

N° US	Theme	En tant que	Je veux...	afin de...	Critères d'acceptance	Priorité	Status	Story Point	Sprint	US accepted (done done)	MoSCoW
1	Preparation	Rédacteur du TB	avoir une structure claire et détaillée pour mon rapport	bien débuter le travail	Le document est préparé avec le template de la HES. La structure contient tous les points à aborder, classés dans un ordre logique	1000	●	3	1	Done Done	Must
2	Recherche	Rédacteur du TB	avoir des exemples de cas d'utilisation de l'IA déjà documentés	sélectionner les cas pertinents	Articles sur les différents Use cases Articles sur l'utilisation de l'IA en médecine Article sur l'utilisation de l'IA pour l'admin	990	●	5	1	Done Done	Must
3	Analyse	Chef de Projet	faire une analyse des besoins	décider lesquels seront intégrés au projet	Pouvoir proposer un diagramme de use cases selon les analyses des cas d'utilisation et les besoins des soignants identifiés au travers des interviews.	980	●	21	2	Done Done	Must
9	Analyse	Spécialiste IA	identifier les LLM existants	les comparer et choisir le plus adéquat pour mon projet	Un choix fondé de LLM pour le projet est fait	970	●	13	3	Done Done	Must
5	Analyse	Chef de Projet	réaliser une analyses des ressources à disposition	motiver mes choix technologiques	La structure des données doit être identifiées et documentée, les ressources financières bien définies et le fonctionnement du dossier patient bien appréhendé.	960	●	8	3	Done Done	Must
6	Analyse	Chef de Projet	identifier les contraintes du projet	m'aider dans mes choix technologiques	Les contraintes sont identifiées, analysés et documentés dans le rapport	950	●	5	4	Done Done	Must
7	Architecture	Développeur	analyser les possibilités matérielles	choisir la meilleure option	choix fondé du matériel pour le projet	940	●	5	4	Done Done	Must
8	Architecture	Développeur	analyser les potentielles architecture	choisir la meilleure option	choix fondé d'architecture	930	●	5	4	Done Done	Must
10	Architecture	Développeur	analyser les langages de programmation et les IDE à disposition	choisir les meilleures options	Faire un choix motivé des langages de programmation et IDE.	920	●	5	4	Done Done	Must
11	Fine-tuning	Spécialiste IA	identifier et analyser les méthodes de fine-tuning	choisir la meilleure option	Faire un choix motivé sur les méthodes de fine-tuning à utiliser pour le projet	910	●	8	4	Done Done	Must
12	Base de données	Développeur	identifier et analyser les méthodes de gestion des données	choisir la meilleure option	Faire un choix motivé du type de DB et de gestionnaire de DB	900	●	3	4	Done Done	Must
14	Préparation	Développeur	mettre en place mon environnement de développement	débuter le développement dans les meilleures conditions	L'environnement de développement est fonctionnel et prêts à débiter le développement	880	●	5	4	Done Done	Must
13	Analyse	Développeur	identifier les risques du projet	développer selon les bonnes pratiques	Avoir une documentation sur les attaques connues de LLM	870	●	5	5	Done Done	Must
15	Fine-tuning	Spécialiste IA	fine-tuner un modèle pré-entraîner avec des données médicales	obtenir de meilleurs résultats dans la génération de texte médical	LLM fine-tuné avec des datasets médicaux donne des résultats concluants	860	●	21	5	Done Done	Must
16	Application	Utilisateur	je veux pouvoir donner un texte contenant des explications sur des soins	vulgarisé et traduit pour le patient	L'utilisateur peut entrer une valeur (en attente use case) dans le champ réservé dans l'application et avoir une réponse qui (en attente use case).	850	●	13	5	Done Done	Must
17	Application	Utilisateur	pouvoir uploader des dossiers patient au format XML	obtenir un résumé regroupant tous les points importants	L'utilisateur peut uploader un fichier XML dans le champ réservé dans l'application et avoir en retour un résumé concis du fichier avec tous les points importants.	840	●	13	5	Done Done	Must
18	Application/Évaluation	Utilisateur	veux être assuré que les réponses obtenues soient correctes	pouvoir utiliser l'application en toute confiance	évaluation des performances du LLM à l'aide de métriques et de vérification par des professionnels de la santé	830	●	21	6		Must

DECLARATION DE L'AUTEUR

Je déclare, par ce document, avoir effectué le travail de Bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et de n'avoir utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de Bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception de mon entreprise, co-proprétaire de ce travail et des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail, et que je cite ci-après : M. Schumacher Michael, M. Manzo Gaetano.

Sierre, le 16 août 2024

Arnaud Michelet

