EASYVISTA

UGA
Université
Grenoble Alpes

RAG Vs LLM fine-tuning

Ilias LEUCHI
Mars à Août - 2024

Superviseur
Ali AIT-BACHIR

M2 SSD, Statistique et Sciences des Données
Université-Grenoble-Alpes

Confidentialité

Les informations (données) contenues sur chacune des pages de ce document constituent des informations confidentielles de la société Easyvista. A réception de ce document, le destinataire accepte de garder confidentielles lesdites informations et de ne reproduire ou divulguer d'aucune manière ces informations à toute autre personne que celles directement responsables de l'évaluation de son contenu, sauf autorisation écrite de Easyvista. Cette obligation de confidentialité ne concerne pas les informations qui (i) étaient connues du destinataire avant la réception du présent document sans restriction aucune, ceci pouvant être attesté par des éléments tangibles; (ii) deviennent accessibles au public sans faute du destinataire; (iii) sont valablement reçues par le destinataire à partir d'une autre source sans restriction aucune (iv) font partie d'une prestation de sous-traitance.

Ce document comprend des informations sur les produits Easyvista qui peuvent être soit améliorées, soit supprimées à la discrétion unique de Easyvista.

Easyvista s'est efforcée d'inclure dans ce document les éléments qu'elle considère comme fiables et adaptés à la demande du destinataire. Ni Easyvista ni ses représentants ne garantissent l'exactitude ou la plénitude des informations fournies. Ce document n'est en conséquence transmis qu'à titre informatif en espérant répondre à vos attentes.

Ni Easyvista ni ses représentants ne seront responsables vis à vis du destinataire ou de ses représentants en raison de l'utilisation des informations fournies. Seul un accord définitif écrit, convenu mutuellement, signé par les représentants autorisés des parties, engagera Easyvista.

Ce document peut vous être transmis au format électronique et/ou au format papier à votre convenance. Si son contenu diffère entre la version papier et la version électronique, SEULE la teneur de la version papier engagera Easyvista. Si vous avez des questions concernant les présentes remarques, nous vous proposons de prendre contact avec votre interlocuteur Easyvista.

Résumé

Avec l'essor des grands modèles de langue (LLM), EasyVista, une entreprise de services informatiques, cherche à intégrer l'intelligence artificielle générative dans ses services à travers un agent conversationnel. Un sujet de recherche a été décidé pour comparer différentes approches de spécialisation des LLM, telles que le RAG (Retrieval-Augmented Generation) et le fine-tuning. L'objectif est de développer un modèle capable de répondre à des demandes spécifiques à un domaine, tout en minimisant les coûts pour l'entreprise. L'étude confronte le RAG, le fine-tuning, ainsi que deux techniques combinatoires : le RAFT (Retrieval-Augmented Fine-Tuning) et le fine-tuning avec RAG. Elle présente également une expérience visant à améliorer les performances du RAG grâce à une classification non supervisée. Avec deux jeux de données de questions-réponses et des techniques de traitement automatique du langage (NLP) pour comparer les textes, les premiers résultats montrent que le RAFT offre la meilleure précision dans les réponses générées. L'approche de clustering appliquée au RAG n'a pas permis d'obtenir de meilleurs résultats que le RAG classique. Aujourd'hui, les avancées en intelligence artificielle progressent rapidement, et de nouvelles méthodes encore plus performantes pourraient voir le jour dans les années à venir. Actuellement, les avancées se concentrent principalement sur l'amélioration du RAG et du fine-tuning, plutôt que sur le développement de nouvelles approches.

Remerciements

Je tiens à remercier M. Ali Ait-Bachir, responsable de l'équipe Lab chez EasyVista, pour son soutien et ses encouragements tout au long de mon stage. Il a consacré de nombreuses heures à discuter de mes progrès et m'a donné de précieux conseils pour mes projets futurs. Son expertise a grandement contribué à la réussite de ce stage et a rendu l'expérience très enrichissante.

Je remercie également le département R&D pour leur accueil et leur convivialité. Une mention spéciale à l'ensemble de mon équipe et à M. Hossein Mohanna, dont les conseils et les contributions ont été essentiels pour la qualité de mon travail.

Contents

1 | Introduction

1.1 | EasyVista

EasyVista est une entreprise spécialisée dans le domaine du logiciel de gestion des services informatiques (ITSM : Information Technology Service Management), et de gestion des opérations informatiques (ITOM : Information Technology Operations Management). Elle s'est rapidement imposée comme un acteur majeur dans le secteur grâce à sa plateforme innovante et ses solutions flexibles qui permettent aux organisations de gérer efficacement leurs services informatiques et leurs processus métier.

La mission d'EasyVista est de simplifier la complexité de l'informatique en fournissant une plateforme entièrement intégrée pour la gestion des services informatiques, permettant aux organisations de maximiser leur valeur commerciale grâce à une informatique accessible.

EasyVista investit continuellement dans l'innovation technologique pour répondre aux défis en constante évolution du marché. Sa plateforme est conçue pour être flexible et adaptable, s'intégrant facilement avec les systèmes existants et s'adaptant aux besoins spécifiques de chaque organisation.

Aujourd'hui, EasyVista est un leader mondial avec plus de 3000 clients dans plus de 150 pays. Elle emploie plus de 400 collaborateurs répartis dans 7 pays. Ses solutions sont conçues pour être évolutives, répondant ainsi aux besoins des petites entreprises comme des grandes.

1.2 | Contexte et objectifs

Le 30 novembre 2022 OpenAI, une organisation de recherche en intelligence artificielle, dévoile leur dernière innovation qui allait bouleverser tous les secteurs. ChatGPT, un agent conversationnel capable de fournir une réponse, rapide, précise et unique, et cela dans divers domaines tel que l'éducation, le développement de logiciels, ou même la médecine. Cette nouvelle technologie révèle le potentiel de l'intelligence artificielle, communément appelé IA, et plus particulièrement l'IA générative. Cette branche de l'IA fait référence aux modèles capable de crée du contenu original, comme du texte, des images, de la musique, contrairement aux systèmes d'IA disciminative qui sont programmés pour effectuer des tâches spécifiques.

L'IA générative est naturellement devenue un enjeu pour de nombreuses entreprises en quête de nouvelles innovations. Cependant, elle présente également des défis importants à surmonter. Pour développer un modèle d'IA de haute qualité, il est nécessaire d'avoir un volume de données important et des composants informatiques performants, dont peu de sociétés disposent. Ce qui implique des difficultés liées au coût des ressources technologiques nécessaires pour la production. Une autre limite réside dans la qualité et la confidentialité des données. Afin d'obtenir des modèles personnalisés et efficaces, les entreprises doivent utiliser des données sensibles. La collecte, le stockage et l'utilisation de ces données doivent être soigneusement gérés pour éviter tous risques associés à la sécurité des informations. L'objectif pour les entreprises est alors de surmonter ces défis. Elles doivent être capables de trouver un équilibre entre l'innovation technologique à faible coût,

et la protection des données privées.

A la recherche de nouvelles technologies informatiques et numériques, EasyVista s'est logiquement orienté vers l'IA, et l'IA générative pour améliorer leur plateforme d'ITSM. L'IA générative peut être utile pour, automatiser les processus de ticketing en catégorisant les demandes en fonction des données et du contexte, une meilleure compréhension du langage naturel pour une meilleure communication, ou encore créer de la documentation basée sur des référentiels de données et de connaissances existants. La recherche en IA générative devient alors un enjeu majeur pour EasyVista.

C'est dans ce contexte que la société a dédié une part de sa recherche au développement d'un agent conversationnel. Pour ce faire, elle s'appuie sur les avancées récentes en traitement automatique du langage naturel, ou NLP (Natural Language Processing), et sur l'utilisation des modèles de langage de grande taille, LLM (Large Language Models). Le NLP est une branche de l'IA qui se concentre sur les interactions entre les ordinateurs et les humains à travers le langage naturel. Les LLM sont des modèles entraînés sur d'énormes volumes de données, ce qui leur permet de générer du texte de manière cohérente et dans un contexte approprié. En combinant les capacités du NLP et des LLM, EasyVista vise à créer un agent conversationnel capable de répondre à des questions spécifiques à leur produit. Cependant, pour répondre précisément à des questions propres à un domaine, une phase d'intégration de connaissances dans le LLM doit être réalisée. En effet, bien qu'un modèle soit entraîné sur un vaste volume de données, il ne possède pas les connaissances internes et spécifiques d'une entreprise. De plus, il est essentiel que les informations provenant d'EasyVista soient exploitables. Un travail de récupération et de prétraitement des données est donc nécessaire. Face à l'ensemble de ces défis, la question suivante se pose : comment obtenir un modèle d'IA génératif, à faible coût, capable de répondre à des questions spécifiques à un domaine, sans menacer la confidentialité des informations privées ?

Pour répondre à cette problématique, deux approches se sont imposées comme des plus prometteuses. Il s'agit de la génération augmentée de récupération, ou RAG (Retrieval-Augmented Generation), et le fine-tuning. Le RAG est une technique qui combine les capacités des modèles de langage de grande taille avec un système de recherche d'information. Lorsque des questions sont posées au LLM, il extrait, d'une base de données spécifique à un domaine, des informations pertinentes. Cela permet de fournir des réponses précises, tout en préservant la confidentialité des informations sensibles, car les données privées restent contrôlées et accessibles uniquement au moment de la requête. D'autre part, le fine-tuning consiste à ajuster un LLM en l'entraînant de nouveau sur un jeu de données spécifique à un domaine. Cette méthode permet d'ajouter des connaissances spécialisées directement dans le modèle, le rendant capable de répondre avec précision à des questions particulières au domaine.

Pour déterminer quelle méthode convient le mieux aux besoins et aux ressources d'EasyVista, un travail de recherche visant à confronter le RAG et le fine-tuning a été décidé. Aujourd'hui, nous présentons les travaux réalisés à cet effet. Notre objectif est d'établir des protocoles de test pour le RAG et le fine-tuning, tout en proposant de nouvelles approches pour améliorer les résultats. Nous débuterons par une définition approfondie des principaux concepts. Par la suite, nous examinerons diverses recherches existantes sur le

RAG, le fine-tuning et d'autres méthodes, et présenterons notre propre travail, basé sur une classification de texte non supervisée, visant à améliorer l'état de l'art. Nous détaillerons les méthodologies adoptées et les données utilisées pour comparer ces différentes approches. Enfin, nous présenterons les résultats obtenus et les conclusions tirées de cette étude.

2 | Définition des principaux concepts

Pour mieux appréhender les prochaines étapes, explorons plus en détail les concepts clés. Nous approfondirons : l'IA générative, le NLP, les LLM, le RAG et le fine-tuning.

2.1 | IA générative

Afin de générer du contenu original et similaire à celui d'un humain, l'IA générative [30] utilise des données existantes pour estimer la probabilité de nouvelles données ou pour capturer leur structure. Les réseaux de neurones artificiels [14] jouent un rôle dans ce processus, étant des modèles inspirés du fonctionnement du cerveau humain. On retrouve une structure composée de neurones artificielles et de couches (voir figure 1). Les neurones artificiels reçoivent des entrées qu'ils transforment à l'aide d'une fonction d'activation et produisent une sortie. Les couches sont constituées d'un ensemble de neurones. On distingue trois niveaux de couches dans un réseau de neurones : la couche d'entrée, qui reçoit les données, la couche de sortie, qui produit le résultat final, et les couches cachées, situées entre l'entrée et la sortie, qui effectuent des transformations intermédiaires. Le nombre de couches et de neurones par couche peut varier selon le problème et/ou les données.

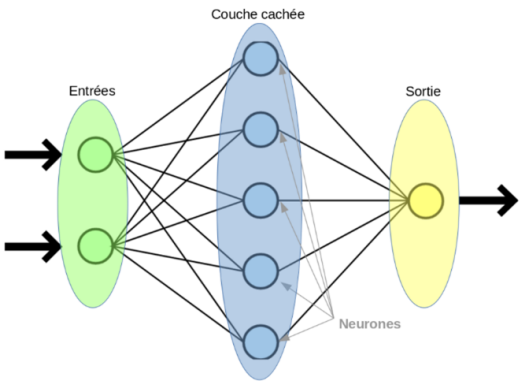

Figure 1: Représentation d'un réseau de neurones

Le fonctionnement d'un réseau de neurones artificiels est le suivant : les données d'entrée traversent les neurones, couche par couche. Chaque neurone effectue une combinaison linéaire des entrées, qui est ensuite transformée par une fonction d'activation, utilisée pour introduire la non-linéarité, et produire la sortie. La différence entre la sortie prédite et la sortie réelle est évaluée à l'aide d'une fonction de perte, ou loss. Les gradients de cette fonction de perte par rapport aux poids du réseau sont calculés, puis ils sont utilisés pour mettre à jour les poids à l'aide d'un algorithme d'optimisation tel que la descente de gradient. Ce processus permet au réseau de neurones d'ajuster ses poids progressivement, le but étant de minimiser la perte et d'améliorer ses performances dans la tâche spécifique pour laquelle il est entraîné.

2.2 | NLP

Le traitement automatique des langues est aujourd'hui au cœur de nombreux sujet d'IA. Il constitue le lien qui permet toute interaction entre l'humain et la machine. Cette tâche comprend plusieurs étapes clés. Tout d'abord, il y a la normalisation des données, où le texte est segmenté en unités appelées tokens, souvent par mots individuels. Ensuite, certaines opérations peuvent être réalisées sur ce texte, telles que la lemmatisation pour déterminer la forme canonique des mots, ou bien la suppression des mots vides. Enfin, il est important de convertir les données textuelles en données numériques à travers un processus appelé embedding. Ces embeddings sont créés à partir de modèles qui encodent le texte de différentes manières. Par exemple, certains modèles utilisent le nombre d'occurrences des mots alors que d'autres capturent le sens des phrases. Il est donc important de choisir un modèle d'embedding adapté à la tâche que nous souhaitons réaliser. La figure 2 illustre les processus de tokenisation et d'embedding.

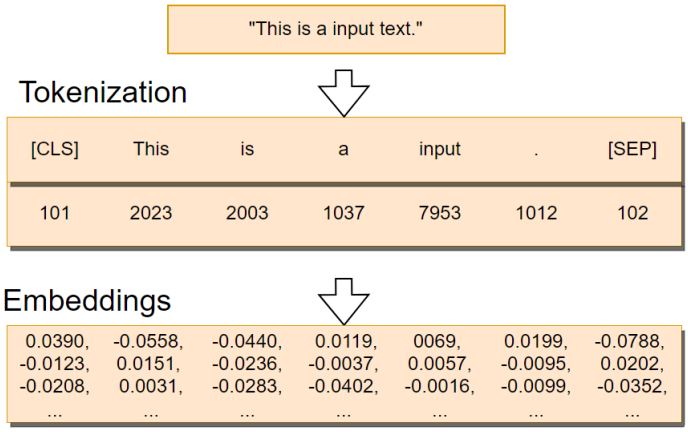

Figure 2: Tokenisation et embeddings [34]

Une machine ne pouvant comprendre que des données numériques, ces traitements du texte sont primordiaux pour le bon fonctionnement des tâches d'IA générative impliquant des données textuelles, telles que la traduction automatique, l'analyse de sentiment, ou encore les agents conversationnels.

2.3 | LLM

Les modèles de langage de grande taille [19] sont conçus pour comprendre et générer du texte. Ils apprennent à prédire des suites de mots en fonction du contexte fourni par l'entrée. Un LLM est caractérisé par son nombre de couches et de neurones, que l'on nomme paramètres du modèle. Plus il contient de paramètres, plus il sera performant, mais cela entraîne également des coûts plus élevés. Ces modèles utilisent une architecture de réseau de neurones particulière appelée Transformer (voir figure 3), qui est efficace pour traiter des séquences de texte longues et pour capturer des dépendances entre les mots. Les Transformers utilisent une architecture encodeur-décodeur. L'encodeur prend la séquence d'entrée et la transforme en une représentation vectorielle. Le décodeur génère la séquence de sortie à partir du vecteur encodé. Les Transformers utilisent également un mécanisme d'auto-attention qui permet au modèle de prendre en compte les relations entre tous les mots simultanément. Ce qui les rend plus efficaces pour des traitements sur de grandes quantités de données. En unifiant l'architecture encodeur-décodeur et le

mécanisme d'auto-attention, cela permet de gérer chaque mot de la séquence en tenant compte de tous les mots précédemment générés.

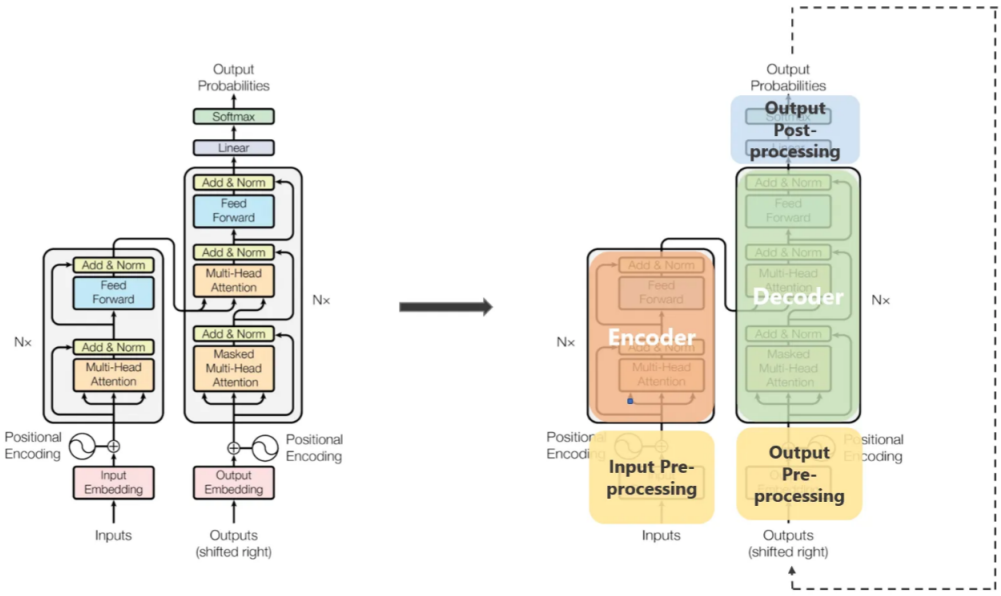

Figure 3: Architecture Transformers [3]

Le développement d'un LLM demande des ressources importantes en terme de données, de matériel, et de temps. Seules les grandes entreprises technologiques, certaines institutions académiques et de recherche, ainsi que des startups bien financées peuvent se permettre de développer ces modèles. À titre d'exemple, le dernier LLM d'OpenAI, GPT-4, a un coût total estimé à 60 millions de dollars pour l'entraînement, sur 13 milliards de tokens, et a nécessité 25 000 cartes graphiques (GPU) pour la phase d'entraînement, qui ont fonctionné pendant 90 jours non-stop [11] .

2.4 | RAG

La génération augmentée de récupération [40] combine la génération à partir d'un LLM avec des techniques de recherche d'information. L'objectif est de fournir un maximum d'informations utiles au modèle, sur un domaine précis, afin de lui permettre de générer une réponse conforme. Le RAG se décompose en trois étapes principales (voir figure 4). Tout d'abord, il est nécessaire de stocker les données complémentaires, qui sont ensuite découpées en morceaux de texte appelés "chunks". Chaque morceau passe par une phase d'embedding. Une fois toutes les informations transformées, elles sont stockées dans une base de données de vecteurs d'embeddings. Ensuite, vient la phase de recherche d'information. Pour cela, il faut d'abord créer un embedding du prompt (instruction donnée au modèle par l'utilisateur). Il suffit ensuite d'effectuer une recherche de similarité entre le prompt et la base de données, afin d'en extraire les documents pertinents pour répondre à la demande de l'utilisateur. Enfin, il faut demander au LLM d'utiliser les informations retrouvées pour répondre au prompt de l'utilisateur.

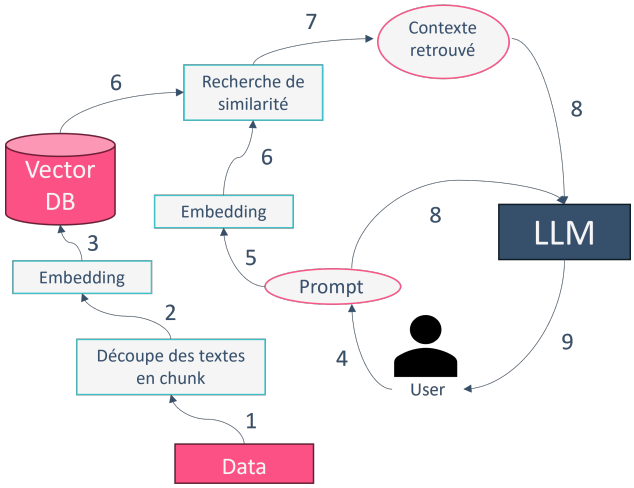

Figure 4: Génération augmentée de récupération (RAG)

Le RAG permet ainsi au LLM d'améliorer sa précision lors de la génération, tout en réduisant le risque de fournir des informations incorrectes, appelées hallucinations du modèle. De plus, le RAG permet au LLM d'intégrer rapidement de nouvelles informations sans nécessiter une phase de réentraînement.

2.5 | Fine-tuning

Le fine-tuning [16] est une méthode plus classique pour apporter du contexte à un LLM. Cette technique consiste à adapter un modèle à une nouvelle tâche spécifique par l'ajout de nouvelles données. Pour fine-tuner un LLM, il faut d'abord choisir le modèle le plus approprié à notre tâche. Chaque LLM a une manière particulière de traiter les informations. Il est donc important de préparer les données en fonction du LLM et de la tâche que nous souhaitons apprendre au modèle. Enfin, nous pouvons ajuster les hyperparamètres du modèle [35] afin de trouver la combinaison qui offrira les meilleurs résultats. Le modèle est ensuite réentraîné sur les nouvelles données spécifiques. La figure 5 illustre le processus de fine-tuning.

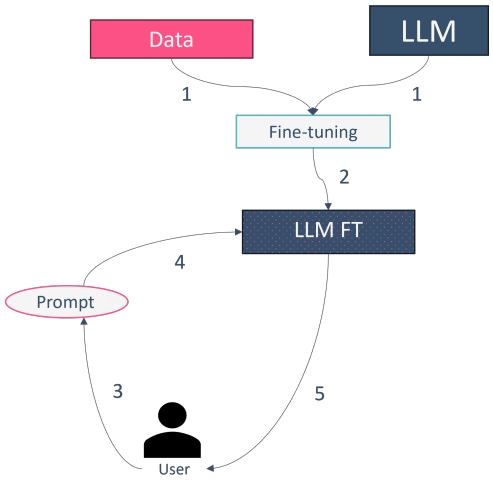

Figure 5: Fine-tuning

Le fine-tuning d'un LLM permet d'améliorer les performances sur des tâches spécifiques tout en conservant ses connaissances passées. Néanmoins, cette tâche reste très gourmande en ressources. C'est pourquoi des techniques d'optimisation des LLM ont été développées.

Deux d'entre elles sont particulièrement utilisées : la quantification et le LoRA (Low-Rank Adaptation), une technique de fine-tuning efficient des paramètres, ou PEFT (Parameter-efficient fine-tuning). La quantification [13] consiste à réduire la précision des poids d'un modèle (voir figure 6). Par défaut, les poids sont représentés par des nombres à virgule flottante de 32 bits. La quantification réduit cette précision en utilisant des représentations à plus faible précision, comme des entiers de 8 bits ou de 16 bits.

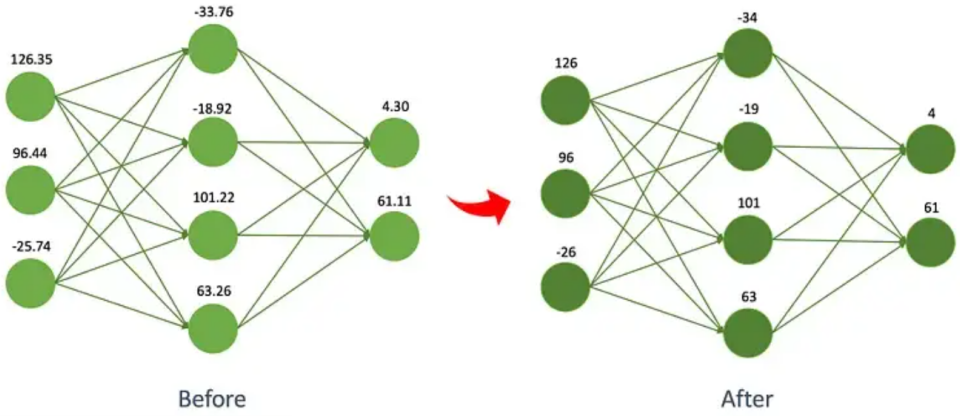

Figure 6: Quantification

Cette technique permet de réduire la taille d'un modèle et d'améliorer son efficacité en termes de mémoire, sans compromettre significativement ses performances.

Le LoRA [33] est une technique qui permet de fine-tuner un modèle sans réentraîner tous ses paramètres. Dans un LLM, les couches sont représentées par des matrices de poids. L'objectif du LoRA est de les décomposer en matrices de rang plus faible, ce qui réduit le nombre de paramètres à optimiser. En combinant le LoRA avec la quantification, nous obtenons une nouvelle technique appelée QLoRA. Celle-ci consiste à d'abord décomposer les matrices de poids, puis à appliquer la quantification sur ces poids (voir figure 7).

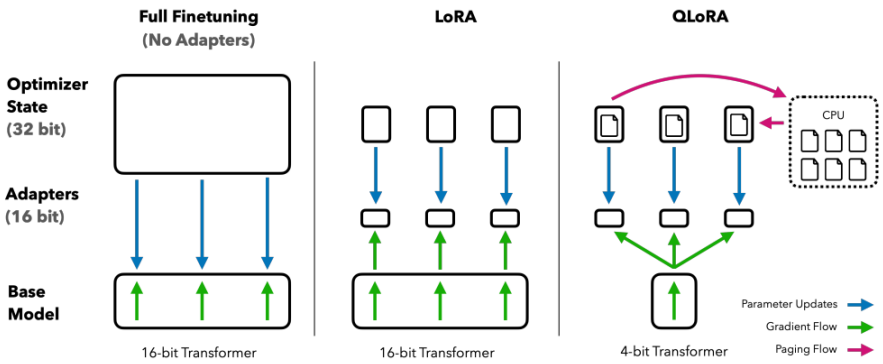

Figure 7: LoRA et QLoRA [33]

Bien qu'elle entraîne une perte de précision plus importante, QLoRA reste l'approche la plus intéressante pour le fine-tuning de LLM. Ceci est principalement dû aux fortes économies de mémoire réalisées.

Les principaux concepts utiles à notre étude étant définis, penchons-nous maintenant sur les recherches déjà réalisées concernant le RAG et le fine-tuning.

3 | Etat de l'art

Grâce aux avancées technologiques des composants informatiques, aux grands volumes de données disponibles, et aux nouveaux algorithmes et modèles, les progrès de l'intelligence artificielle ont explosé au cours de cette décennie. Naturellement, de nouvelles recherches ont vu le jour, et celles-ci connaissent une évolution fulgurante. Passons en revue quelques articles concernant les LLM, le RAG ainsi que le fine-tuning. Nous parlerons également d'une nouvelle méthode capable d'améliorer les résultats du RAG et du fine-tuning.

Conjointement aux avancées en intelligence artificielle, le nombre de LLM a considérablement augmenté. L'article [36] présente une étude approfondie sur l'évolution de ces modèles.

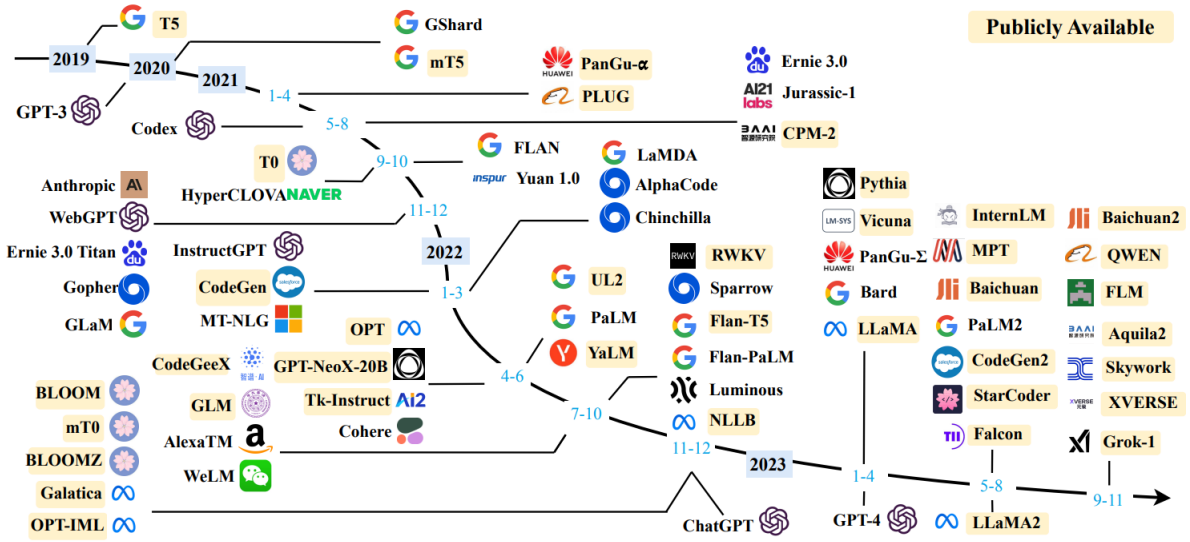

Figure 8: Chronologie des LLM d'une taille supérieure à 10B [36]

Sur la figure 8, on voit la chronologie des modèles de langage de grande taille avec plus de 10 milliards de paramètres. On remarque que de plus en plus de LLM ont vu le jour, mais ceci n'est qu'une partie des modèles existants. En effet, dû au nombre de paramètres des LLM qui ne cesse d'augmenter, de nombreuses organisations se sont concentrées sur le développement de modèles avec un nombre de paramètres réduit, inférieur à 10 milliards. Ces plus petits modèles sont moins précis que les modèles plus grands, mais sont utiles pour des tâches d'ajustement, telles que le RAG et le fine-tuning.

Par exemple, on trouve le modèle Mistral 7B, un LLM de 7 milliards de paramètres conçu par la start-up française Mistral AI, spécialisée dans l'IA générative. Dans l'article [1], il est démontré que ce modèle peut offrir des performances équivalentes, voire meilleures, que d'autres LLM avec plus de paramètres.

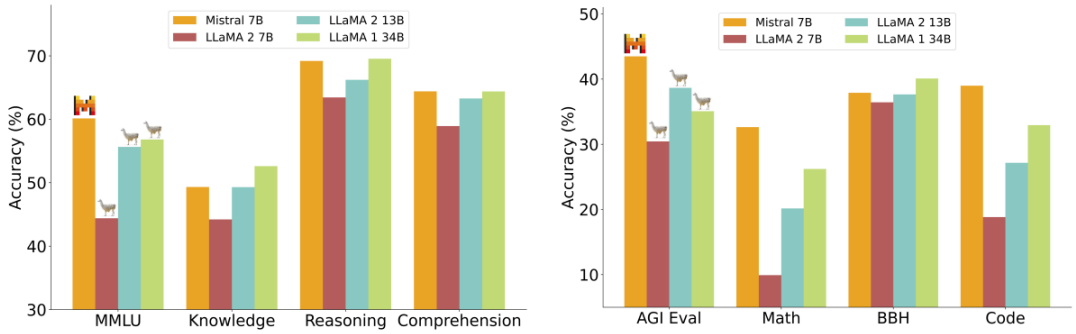

Figure 9: Performances du Mistral 7B et de différents modèles LLaMA sur une large gamme de critères de référence [1]

Dans la figure 9, le modèle Mistral 7B est comparé aux modèles LLaMA de Meta. On se rend alors compte des capacités du modèle à obtenir de bonnes performances. Par exemple, on voit que Mistral 7B est meilleur en mathématiques, en génération de code et en raisonnement que LLaMA 1 34B, un modèle de 34 milliards de paramètres. Une des limites de ce modèle est qu'il est entraîné uniquement avec des textes en anglais et n'est donc pas multilingue.

L'utilisation de LLM à paramètre réduit est très utile pour réaliser davantage d'économie en ressources lors de l'ajustement d'un modèle. Regardons cela avec Le RAG. Cette technique a prouvé son efficacité pour améliorer les réponses des LLM de grandes et de petites tailles. L'article [17] compare des réponses générées avec et sans RAG, en utilisant GPT-4, un modèle de grande taille, et Mistral 7B. Il évalue la qualité des réponses en mesurant leur concordance avec une réponse de référence sur divers sujets.

Mistral-7B	Concordance (Prior)	Concordance (w/ RAG)	Slope
Drug Dosage	0.550	0.677	-0.82
Sports Stats	0.240	0.057	N/A
Latest News	N/A	N/A	N/A
Wikipedia Dates	0.080	0.840	-0.77
Wikipedia Names	0.065	0.760	-0.14
Wikipedia Locations	0.490	0.690	-0.030
Average	*0.285*	*0.605*	*-0.44*

Figure 10: Concordance des réponses générées par Mistral 7B, avec et sans RAG, sur divers sujets [17]

Dans le tableau de la figure 10, nous observons les résultats obtenus avec le modèle Mistral 7B. Globalement, l'utilisation du RAG a permis d'obtenir des réponses plus alignées avec la réponse de référence. Le score moyen de concordance est de 0.605 avec RAG, contre 0.285 sans RAG.

Pour le fine-tuning, l'utilisation de LLM à paramètre réduit est également recommandée. Combiné avec QLoRA, l'ajustement d'un modèle à coût réduit devient une solution raisonnable pour les petites entreprises. L'article [22] est un guide pour le fine-tuning de modèles de langage pour les entreprises. Utilisant les modèles LlaMA 2 de Meta, il

présente les étapes à suivre pour affiner un LLM. Cela inclut le prétraitement des données, l'utilisation des méthodes PEFT, la quantification, et d'autres recommandations. Par exemple, pour la préparation des données, il propose de former des paires de questions-réponses à partir des documents. L'article présente également des résultats en termes de temps d'inférence et de ressources mémoire, afin d'analyser les effets de la compression des LLM. Il montre que la quantification augmente le temps d'inférence, mais réduit fortement la mémoire nécessaire. Il présente aussi les résultats obtenus en combinant le RAG et le fine-tuning. En effet, utiliser un modèle fine-tuné en ajoutant du contexte grâce au RAG est une option à considérer pour spécialiser un modèle. L'article montre l'efficacité de cette méthode dans la génération de réponses, par rapport à une approche RAG classique.

Malgré leur popularité, le RAG et le fine-tuning détiennent leur lot de contraintes. Pour le fine-tuning, le modèle est limité à son ensemble de données d'entraînement, ce qui le rend également vulnérable à l'approximation et à l'hallucination. Quant au RAG, les documents sont sélectionnés uniquement en fonction de leur proximité sémantique avec la requête. Ainsi, le modèle ne peut pas distinguer les documents réellement pertinents de ceux qui ne le sont pas. Ce qui ouvre la porte à d'autres techniques de spécialisation de modèle.

Une méthode a retenu notre attention. Il s'agit du fine-tuning en fonction de l'extraction, ou RAFT (Retrieval-Augmented Fine-Tuning). L'article [32] présente cette méthode. Il s'agit également d'une combinaison entre le RAG et le fine-tuning, la différence se situe au niveau du réentraînement. Ici, les données sont traitées différemment et sont adaptées pour recevoir un contexte provenant du RAG.

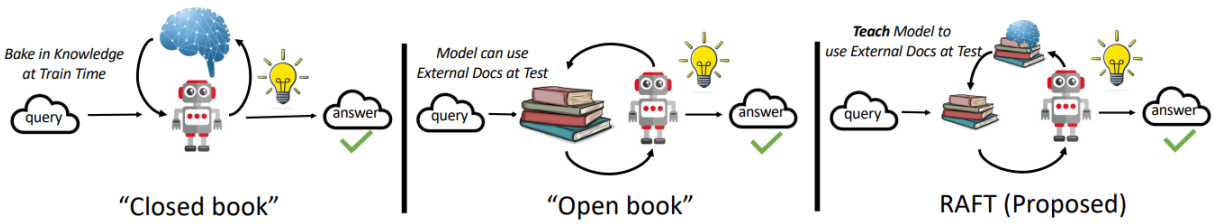

Figure 11: Comparaison des méthodes d'ajustement à un examen [32]

L'article compare les méthodes de spécialisation d'un modèle à un examen (voir figure 11). Le fine-tuning pourrait être vu comme un examen à livre fermé, où il faut acquérir les connaissances avant de répondre aux questions. Le RAG serait un examen à livre ouvert, où il faut chercher la réponse dans un document avant de répondre. Tandis que le RAFT unifie les deux et peut être vu comme un examen à livre ouvert, où les connaissances sont mémorisées avant de répondre.

Une autre particularité du RAFT réside dans les connaissances fournies avant l'entraînement. En effet, lors de la construction du jeu de données d'entraînement, le contexte est fourni de deux manières différentes.

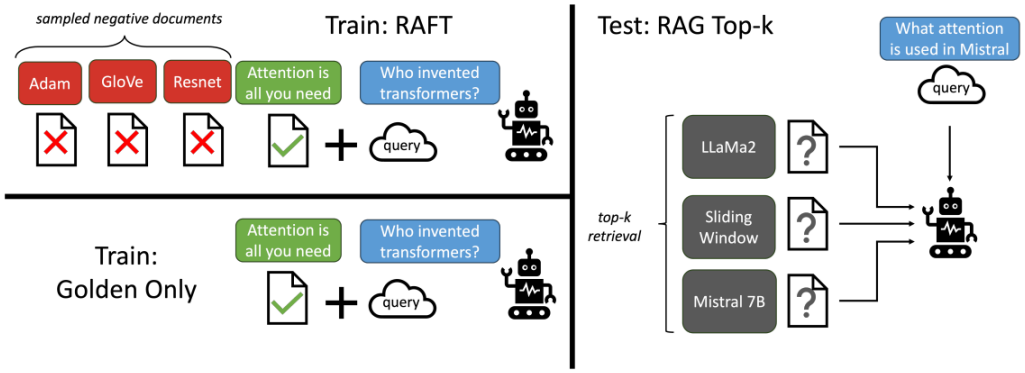

Figure 12: Mise en forme des données pour l'entraînement du RAFT [32]

La figure 12 illustre cette idée. Le jeu de données d'entraînement se compose alors d'une première partie composée des questions, des contextes et des réponses. Ensuite, on ajoute les mêmes questions et réponses, mais avec un contexte saturé de documents perturbateurs. Le modèle apprend ainsi à identifier les documents peu pertinents, et à se concentrer sur le contexte utile pour répondre à une question.

Le RAFT est ensuite comparé aux autres méthodes de spécialisation de LLM, sur différents domaines. L'article utilise le modèle LLaMA 2 7B pour réaliser ses tests. Il examine aussi les résultats du RAG avec le modèle GPT-3.5, un des LLM les plus performants du moment, utilisé notamment pour ChatGPT. Il évalue ensuite la similarité entre les réponses générées et des réponses de référence.

	PubMed	HotPot	HuggingFace	Torch Hub	TensorFlow
GPT-3.5 + RAG	71.60	**41.5**	29.08	60.21	65.59
LLaMA2-7B	56.5	0.54	0.22	0	0
LLaMA2-7B + RAG	58.8	0.03	26.43	08.60	43.06
DSF	59.7	6.38	61.06	84.94	86.56
DSF + RAG	71.6	4.41	42.59	82.80	60.29
RAFT (LLaMA2-7B)	**73.30**	35.28	**74.00**	**84.95**	**86.86**

Figure 13: Résultats des différentes méthodes de spécialisation de LLM, sur divers domaines [32]

La figure 13 présente les résultats obtenus. Ici, DSF correspond au fine-tuning. On observe que le modèle RAFT fournit les meilleures réponses dans 4 des 5 domaines testés. Le RAFT représente donc une option prometteuse à explorer sur nos propres données. Une des limites du RAFT réside dans la quantité de données nécessaires. En effet, cette technique exige un réentraînement sur un plus grand volume de données textuelles, ce qui augmente inévitablement les coûts de calcul.

Ce passage en revue des LLM, du RAG, du fine-tuning et d'autres approches nous permet de prendre conscience des bonnes pratiques à adopter pour notre étude. Nous avons également cherché à apporter notre propre idée dans le but d'améliorer l'état de l'art. Regardons maintenant notre proposition.

4 | Contribution (clustering pour le RAG)

Pour améliorer les recherches actuelles, nous avons tenté d'utiliser la classification de texte non-supervisée. Cette approche est utilisée pour organiser des textes en groupes sans utiliser des données étiquetées. L'objectif est alors de découvrir des structures dans les données permettant d'identifier les textes traitant d'un même sujet. Pour le RAG, nous avons perçu le potentiel du clustering de texte afin d'aider le modèle à mieux cibler les demandes des utilisateurs.

L'idée est la suivante (voir figure 14) : il faut d'abord réaliser le clustering des morceaux de texte afin que chacun soit associé à un cluster. L'information du cluster est ensuite stockée dans les métadonnées. Par la suite, un cluster est attribué au prompt de l'utilisateur. Cette information est alors utilisée pour la recherche de similarité. Le cluster du prompt peut être appliqué pour filtrer les chunks retrouvés par la recherche sémantique, par exemple en ne conservant que les morceaux de texte appartenant au même cluster. Les documents sélectionnés peuvent alors être employés pour le RAG.

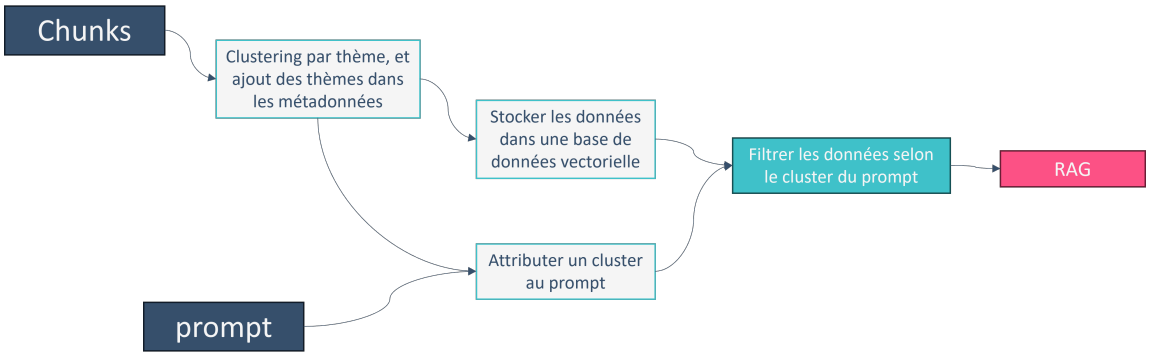

Figure 14: Clustering pour le RAG

Si cette méthode s'avère efficace, elle pourrait être utile pour toute autre technique s'appuyant sur le RAG, telle que le RAFT.

C'est ici que s'achève notre tour des idées pertinentes permettant de répondre à notre problématique. Regardons maintenant les méthodologies retenues ainsi que les données utilisées pour l'évaluation.

5 | Données et méthodologies d'évaluation

Dans cette section, nous présentons les deux jeux de données utilisés pour nos analyses. Nous détaillons les outils employées pour réaliser le RAG et le fine-tuning, ainsi que les métriques utilisées pour évaluer les textes générés. Nous présentons les techniques de clustering mises en place et terminons avec une particularité propre à notre contribution.

5.1 | Description des jeux de données

L'obtention de jeu de données d'évaluation est une étape importante dans ce travail. Pour confronter le RAG et le fine-tuning, nous avons retenu l'approche de données composées de pair de questions-réponses. EasyVista est une entreprise spécialisée dans l'IT (Information

Technology), il est donc essentiel d'avoir des données traitant d'informatique. Regardons les deux jeux de données que nous avons construit.

5.1.1 | Données Yahoo

Notre point de départ est un ensemble de données questions-réponses (Q/A) open source fourni par Yahoo, qui classe les questions en 10 catégories [39]. Nous ne conservons que les données appartenant à la catégorie 5, correspondant à la classe "Computers and Internet". Ce sous-ensemble est ensuite prêt pour le fine-tuning. Pour le RAG, nous appliquons une classification non supervisée à ces données afin de regrouper les paires questions-réponses par thème. L'idée est de regrouper les réponses d'un même thème pour simuler des documents, qui seront ensuite stockés dans la base de données vectorielle.

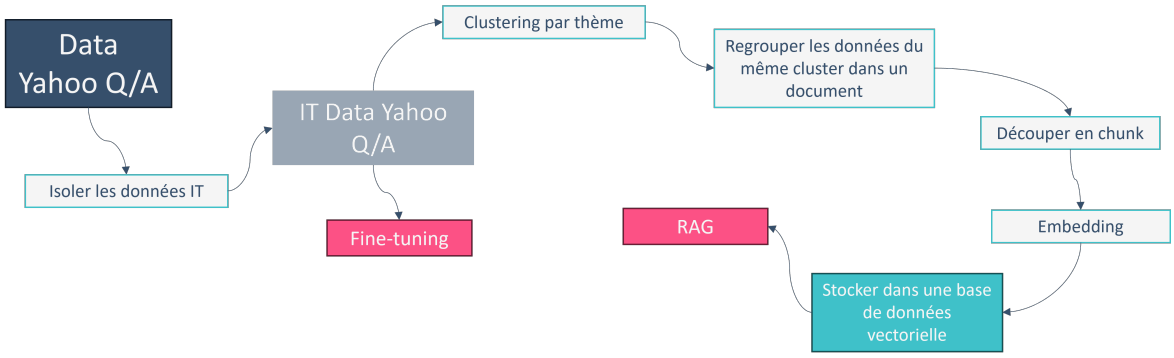

Figure 15: Prétraitement des données Yahoo

La figure 15 illustre le prétraitement des données Yahoo. Nous obtenons alors un total de 5736 données.

5.1.2 | Données Service Manager

Pour faciliter l'utilisation de ses services, EasyVista a mis en place un wiki qui centralise toute la documentation de ses produits [37]. EV Service Manager (SM) est l'outil de gestion des services IT développé par EasyVista. Nous avons décidé de créer un jeu de données à partir de la documentation de ce produit.

Pour ce faire, nous avons commencé par extraire toutes les pages web du wiki Service Manager. Étant donné que le contenu est au format HTML, une conversion en texte brut est nécessaire. Cette transformation préserve la structure de certains éléments HTML, tels que les listes et les tableaux, et retire les images.

Ensuite, nous découpons le texte en chunks et utilisons ces morceaux pour deux tâches. La première, classique au RAG, consiste à stocker les chunks dans une base de données vectorielle. La deuxième tâche est utile à l'évaluation des réponses et au fine-tuning. Nous générons alors trois paires questions-réponses pour chaque chunk à partir du modèle Mistral 7B. Cela nous permet de créer un jeu de données pour le fine-tuning et de vérifier si le RAG arrive à retrouver le chunk associé à la question. Les réponses générées serviront également de référence.

Si nous divisions le jeu de données en données d'entraînement et de test, nous risquerions de perdre des informations pendant le fine-tuning. Pour conserver toutes les connaissances,

nous utilisons un jeu de données distinct pour l'évaluation. Ce jeu de données de test est composé de questions générées à l'aide du modèle LLaMA 3 8B, à partir de 300 chunks aléatoires, mais avec une seule question par chunk. Nous nous assurons alors des résultats non biaisés.

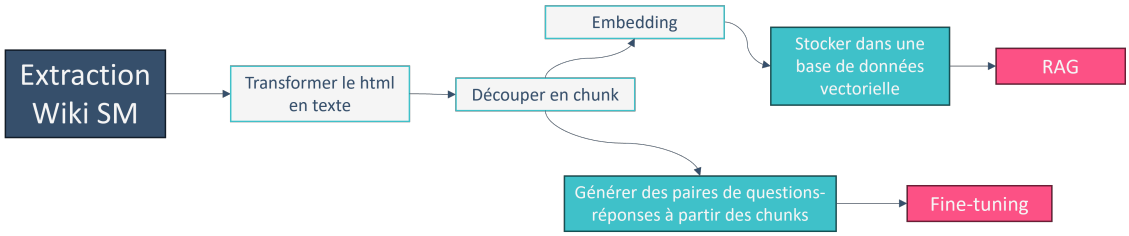

Figure 16: Prétraitement des données Service Manager (SM)

La figure 16 illustre le prétraitement des données SM. Nous disposons de 8010 paires de questions-réponses pour le jeu de données principal, et de 300 paires pour le jeu de données de test.

5.2 | Packages

Pour effectuer les tâches de RAG et de fine-tuning, nous utilisons des packages spécialisés. Plus précisément, nous utilisons LangChain [2] pour le RAG et à Unsloth [10] pour le fine-tuning. Voyons plus en détail ces deux packages.

Le package LangChain est très utile pour le RAG. Il permet de simplifier l'utilisation des LLM, d'effectuer la recherche de similarité, de faciliter la création de bases de données vectorielles, et de découper les documents en chunks. L'article [38] montre les meilleures pratiques pour le RAG. Regardons les choix que nous avons faits. Pour la recherche de similarité, nous avons choisi de retrouver les 5 documents les plus proches du prompt. Les documents sont découpés en chunks de 2000 caractères maximum, avec un chevauchement entre les morceaux de 200 caractères. Avec les données SM, cela représente 2714 chunks, avec une moyenne de 345 tokens. Pour la création des bases de données, nous avons utilisé FAISS [23] et Chroma [6], deux outils populaires pour la gestion de données vectorielles. Le modèle d'embedding utilisé est "all-mpnet-base-v2", un modèle de Sentence-BERT [26, 31] adapté à de nombreux cas d'utilisation, qui prend un maximum de 384 tokens en entrée. La recherche des meilleurs paramétrages pour le RAG est l'objet d'un autre sujet de recherche chez EasyVista.

Le package Unsloth facilite le fine-tuning de LLM en utilisant plusieurs techniques d'optimisation, telles que la quantification et le LoRA. Grâce à ces méthodes, Unsloth permet de gagner en rapidité et d'optimiser l'utilisation de la mémoire. C'est pourquoi nous avons choisi d'utiliser ce package pour nos tâches de fine-tuning.

Langchain et Unsloth utilisent tous deux des fonctionnalités de Hugging Face [12], une start-up spécialisée dans l'intelligence artificielle. Hugging Face facilite l'accès au domaine de l'IA en regroupant plusieurs outils importants, tels que des bibliothèques, des modèles, et bien plus encore.

5.3 | Métriques d'évaluation

Une des étapes primordiales de notre étude est l'évaluation des réponses générées par nos modèles. Pour ce faire, nous utiliserons les questions et les réponses de nos deux jeux de données. Les questions sont des simulations des demandes utilisateur. Nous comparerons alors le texte généré avec les réponses aux questions, qui font office de référence. Deux métriques ont été retenus pour l'évaluation :

- **Similarité cosinus** [20] **:** mesure le cosinus de l'angle entre deux vecteurs d'embedding, ce qui donne une indication de leur similitude. Retourne une valeur entre -1 et 1. Plus cette valeur est proche de 1, plus les textes sont similaires, et inversement.
- **ROUGE-1** [5] **:** ROUGE (Recall-Oriented Understudy for Gisting Evaluation) est un ensemble de métriques d'évaluation de texte. ROUGE-1 est une des variantes des métriques ROUGE. Elle calcule le rappel (recall) des unigrammes (séquences d'un mot) communs entre deux textes. ROUGE-1 est alors la proportion de mots communs entre les deux séquences de texte.

$$\text{ROUGE-1} = \frac{\text{Nombre de mots communs}}{\text{Nombre de mots total dans le texte de référence}}$$

La similarité cosinus évalue davantage le sens des textes, tandis que le ROUGE-1 se concentre sur les mots. Bien que les deux métriques soient pertinentes, nous privilégions une similarité cosinus élevée pour l'évaluation de nos textes.

5.4 | Méthodes de clustering

Pour réaliser le clustering de nos données, nous avons comparé les modèles d'embedding "all-mpnet-base-v2" et "all-MiniLM-L6-v2", ce dernier étant un modèle qui prend en charge des séquences d'entrée plus courtes. Nous avons conservé le modèle "all-MiniLM-L6-v2" car il offre un gain de temps de calcul, pour une précision légèrement inférieure. Après avoir généré les embeddings de nos textes, nous avons réduit leur dimensionnalité à un espace exploitable en utilisant UMAP [18], une technique de réduction de dimensionnalité qui conserve la structure des données textuelles. Une dimensionnalité trop faible peut entraîner une perte d'information, tandis qu'une dimensionnalité trop élevée peut nuire aux résultats de clustering. Pour trouver un bon équilibre, nous avons réduit à 5 dimensions. Nous avons ensuite comparé deux méthodes de classification non supervisée du texte :

- **HDBSCAN** [24, 21] **:** extension de DBSCAN (un algorithme de clustering basé sur la densité) qui intègre les avantages des algorithmes de clustering hiérarchique. HDBSCAN produit rapidement de bons résultats avec très peu de paramètres à ajuster. Par exemple, il n'est pas nécessaire de spécifier le nombre de clusters à l'avance. De plus, HDBSCAN offre la possibilité de définir un nombre minimum de points pour qu'un cluster soit valide, et permet également d'identifier des points qui ne correspondent à aucun cluster, les considérant comme des valeurs aberrantes. Une des limites de HDBSCAN est qu'il peut devenir coûteux en termes de temps de calcul avec un très grand volume de données.
- **K-means** [15] **:** permet de regrouper un ensemble de données en un nombre fixe de clusters, appelé k. La procédure est la suivante : on commence par choisir k, le

nombre de clusters souhaité. Une sélection aléatoire de k points sera faite, ceux-ci feront office de centres des clusters (centroïdes). Ensuite, pour chaque point de données, on calcule la distance par rapport aux centroïdes, et on assigne le point au cluster avec le centroïde le plus proche. Une fois que tous les points ont été assignés à des clusters, on recalcule les centroïdes. Ce processus est répété avec les nouveaux centroïdes jusqu'à ce que les clusters deviennent stables. K-means est un algorithme assez répandu, qui est rapide et facile à implémenter. Un des principaux inconvénients est qu'il clusterise toutes les données, ce qui force le regroupement de certains points.

Les deux algorithmes utilisent des éléments aléatoires dans leur fonctionnement. Nous testons alors différentes valeurs de "seed", qui permettent de contrôler l'aspect aléatoire, et nous conservons celle qui offre les meilleurs résultats.

Pour évaluer nos clusters, nous utiliserons le coefficient de silhouette [28] , qui mesure la qualité de la séparation entre les clusters. Cette métrique varie entre -1 et 1. Une valeur proche de 1 indique que les clusters sont bien séparés et que les points sont bien regroupés au sein de leur propre cluster, et inversement.

Pour HDBSCAN, nous avons imaginé un indicateur visant à trouver un équilibre entre le nombre de données aberrantes et la qualité des clusters. Nous cherchons alors à réduire le nombre de valeurs aberrantes tout en maximisant le coefficient de silhouette. Pour cela, nous calculons les deux indicateurs en utilisant différentes valeurs de "seed" pour obtenir deux échantillons. Ces échantillons sont ensuite normalisés (Min-Max Scaling), puis nous effectuons le calcul suivant :

$$\alpha \times \text{coeff. Silhouette normalisé} - \beta \times \text{nb valeur aberrante normalisé}$$

Nous cherchons à maximiser cette valeur. Les paramètres α et β permettent de gérer le poids de chaque indicateur. Un α élevé accorde plus d'importance à des clusters bien regroupés, tandis qu'un β élevé favorise un faible nombre de valeurs aberrantes. Dans notre cas, nous privilégions des clusters bien regroupés en fixant α à 1.5 et β à 1.

5.5 | Clustering pour le RAG

Une particularité propre à notre apport est à souligner. Nous avons dû choisir la méthode la plus adaptée pour identifier le cluster du prompt. Nous avons testé plusieurs approches et présenterons les deux suivantes :

- **Machine learning :** à partir des embeddings et des clusters de chaque chunk, nous entraînons un modèle de machine learning afin de prédire le cluster du prompt de l'utilisateur. Nous avons testé plusieurs algorithmes de classification en utilisant PyCaret [29], un package Python qui facilite ces tests, et avons retenu l'approche des K plus proches voisins (KNN) [25], une méthode qui regroupe les données en fonction de leur proximité.
- **Similarité cosinus :** cette méthode consiste à calculer la similarité cosinus entre le prompt et chaque chunk, et ainsi conserver le cluster du chunk le plus similaire.

Pour choisir la meilleure approche, nous avons évalué en utilisant 1000 questions aléatoires du jeu de données SM. Nous avons considéré les questions comme des prompts. Les bons clusters étaient ceux des chunks de référence. Nous avons ensuite calculé le taux de bonnes prédictions.

Par la suite, nous avons testé différentes méthodes pour intégrer le cluster dans la recherche sémantique. Celle qui a donné les meilleurs résultats consiste à effectuer la recherche sémantique de manière classique pour retrouver 15 chunks. Si le cluster prédit est -1 (cluster des valeurs aberrantes), nous conservons les 5 premiers chunks. Sinon, nous sélectionnons les 3 premiers documents et les complétons avec 3 chunks provenant du même cluster que le prompt. Ainsi nous nous retrouvons avec 6 documents comme contexte pour le RAG.

La présentation des méthodes employées pour mener à bien cette étude touche à sa fin. Nous allons maintenant regarder et analyser les résultats obtenus afin de déterminer quelles approches ont été les plus efficaces pour spécialiser un LLM.

6 | Analyse des résultats

Notre étude nous a permis d'obtenir les premières idées pertinentes à explorer dans le cadre du développement d'un agent conversationnel spécialisé. Regardons les résultats obtenus. Les données Yahoo nous permettront de choisir le LLM à adapter ainsi que la méthode de clustering à retenir. Par la suite, nous utiliserons les données SM pour confronter les méthodes de spécialisation de modèle.

6.1 | Choix du LLM

La première étape consiste à choisir le bon LLM capable de fournir des réponses correctes avec un nombre de paramètres réduit. Pour cela, nous avons confronté quatre LLM populaires : le LLaMA 3 8B, le Mistral 7B, le Gemma 2B et le Falcon 7B. Pour les comparer, nous simulerons une tâche RAG avec les questions des données Yahoo. Nous utiliserons les réponses pour les comparer à celles des modèles, en calculant la moyenne des similarités cosinus et des ROUGE-1.

	LLaMA3 8B	**Mistral 7B**	**Gemma 2B**	**Falcon 7B**
Similarité cosinus	0.467	**0.499**	0.297	0.275
ROUGE-1	0.237	**0.403**	0.291	0.168

Table 1: Scores de similarité pour différent LLM, avec les données Yahoo

Le tableau 1 nous montre les résultats obtenus. Nous remarquons que le modèle Mistral 7B est celui qui fournit les réponses les plus en accord avec les réponses de référence sur les deux métriques. La similarité cosinus est d'environ 0.5. Lorsque nous observons uniquement la similarité cosinus, le modèle LLaMA 3 8B arrive en deuxième position, avec un score de 0.46, mais ce modèle fournit des résultats nettement inférieurs pour le ROUGE-1. Les modèles Gemma 2B et Falcon 7B obtiennent des scores globalement inférieurs. Nous conserverons donc le modèle Mistral 7B comme LLM pour nos expériences.

6.2 | Méthodes de clustering

Il nous faut à présent choisir la meilleure approche entre HDSCAN et K-means pour réaliser le clustering des données. Pour cela, nous avons clusterisé les questions des données Yahoo. Observons les résultats obtenus avec les deux méthodes.

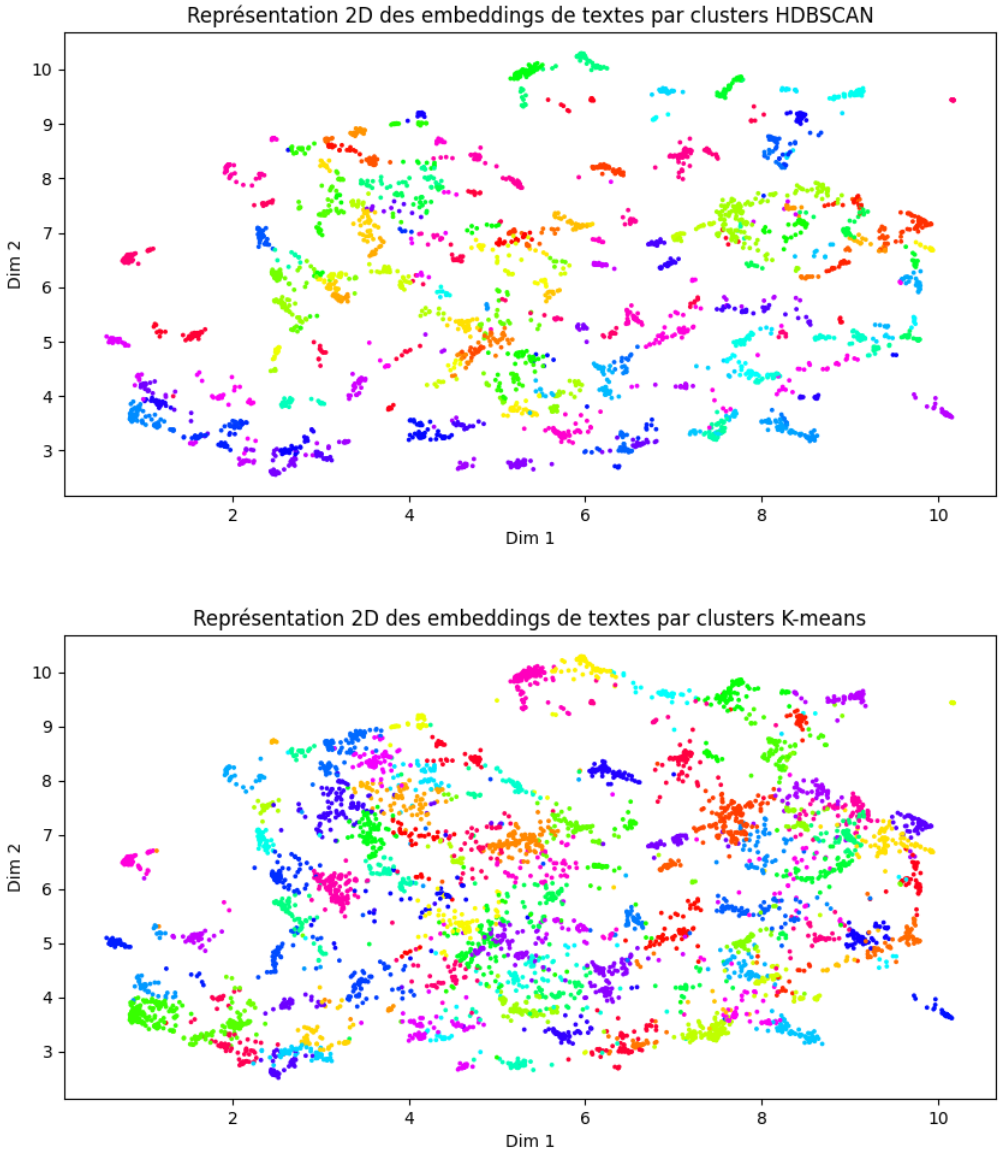

Figure 17: Représentation 2D des embeddings de textes par clusters, avec les données Yahoo, selon la méthode de clustering

Pour visualiser le clustering, nous réduisons les embeddings des textes à deux dimensions. La figure 17 illustre le plan principal, où chaque point représente une question et sa couleur indique le cluster auquel elle appartient. Un graphique est généré pour chaque méthode de clustering. On observe qu'avec HDBSCAN, les clusters sont mieux séparés, tandis qu'avec K-means, ils semblent plus mélangés. Il y a moins de points avec HDBSCAN car moins de questions ont été clusterisées, certaines étant considérées comme des valeurs aberrantes. Alors que K-means clusterise toutes les questions, ce qui explique ce mélange. K-means parvient tout de même à bien identifier les clusters isolés. Regardons les coefficients de

silhouette pour confirmer nos résultats.

	HDBSCAN	K-means
Coeff. Silhouette	0.595	0.417

Table 2: Coefficient de silhouette pour chaque méthode de clustering, avec les données Yahoo

Grâce au tableau 2, nous en déduisons que les clusters sont mieux séparés avec HDBSCAN, car le coefficient de silhouette est supérieur à celui de K-means. Nous choisissons donc HDBSCAN comme méthode de clustering.

Observons maintenant les résultats du clustering appliqué aux chunks des données SM, qui seront utiles pour notre contribution.

	HDBSCAN
Nb clusters	65
Coeff. Silhouette	0.678
Nb valeurs abérantes	628

Table 3: Résultats du clustering HDBSCAN sur les données SM

Le tableau 3 montre que les résultats sont très satisfaisants. Nous obtenons 65 clusters avec un coefficient de silhouette de 0.678 et un total de 628 valeurs aberrantes sur les 2714 chunks au total. Ce nombre de valeurs aberrantes peut sembler élevé, mais il représente le meilleur compromis lorsque nous privilégions une meilleure séparation des clusters.

6.3 | Méthodes de spécialisation

Passons maintenant aux résultats de notre étude principale. Nous avons comparé quatre méthodes pour adapter un LLM : le RAG, le fine-tuning, le RAFT et le fine-tuning associé au RAG. Pour ces quatre approches, nous avons utilisé le modèle Mistral 7B et comparé les réponses générées avec celles du jeu de données SM test, en calculant les moyennes des similarités cosinus et des scores ROUGE-1. Voyons les résultats.

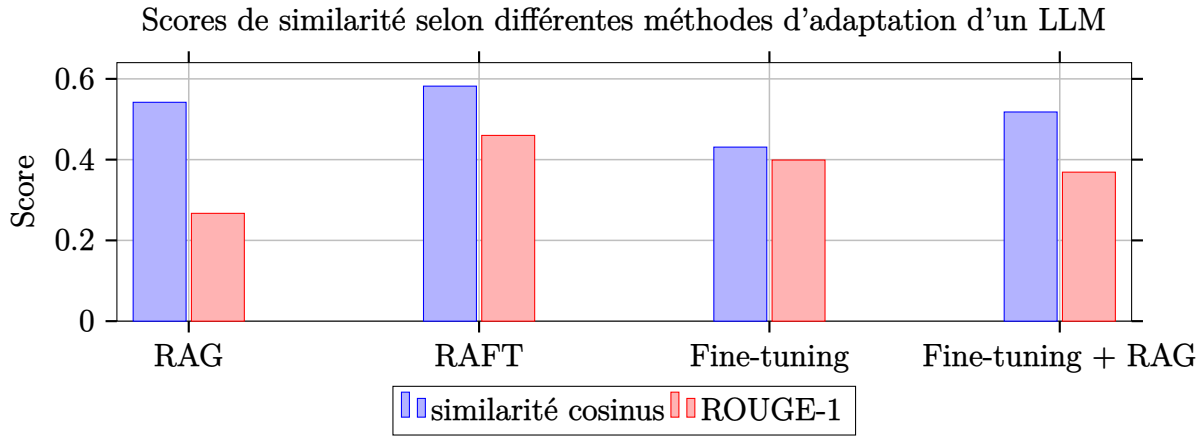

Figure 18: Diagrammes en barres des scores de similarité pour différentes méthodes d'adaptation de LLM, avec les données SM

Sur la figure 18, on remarque que le RAFT obtient les meilleurs résultats pour les deux métriques. En observant uniquement la similarité cosinus, le RAG se classe en deuxième position, juste devant le fine-tuning associé au RAG. En revanche, avec le ROUGE-1, le fine-tuning occupe la deuxième place, tandis que le RAG se retrouve en dernière position.

	RAG	**RAFT**	**Fine-tuning**	**Fine-tuning + RAG**
Similarité cosinus	0.542	**0.582**	0.431	0.518
ROUGE-1	0.267	**0.460**	0.399	0.369

Table 4: Scores de similarité pour différentes méthodes d'adaptation de LLM, avec les données SM

Le tableau 4 nous apporte les résultats chiffrés. Avec une similarité cosinus de 0.582 et un ROUGE-1 de 0.460, le RAFT surpasse les trois autres méthodes.

Ces résultats montrent que le modèle fine-tuned est plus efficace pour saisir la structure des réponses et identifier les termes pertinents pour répondre à une question. Tandis que le RAG excelle dans la compréhension du contexte fourni et la formulation d'une réponse cohérente. L'idée de combiner le fine-tuning et le RAG semble donc très prometteuse. Les résultats obtenus avec RAFT et le fine-tuning couplé au RAG démontrent que cette approche combinée permet d'obtenir de meilleurs résultats. Avec cette étude, le RAFT s'impose comme la meilleure alternative pour spécialiser un LLM.

6.4 | Clustering pour le RAG

Pour terminer notre étude, analysons les résultats obtenus avec notre technique de clustering pour la recherche sémantique du RAG. D'abord en déterminant la meilleure approche pour le clustering d'un prompt, que nous évaluons avec 1000 questions-réponses aléatoires des données SM.

	Taux bonne pred.	**Temps (s)**
Machine Learning	0.609	18.4
Similarité cosinus	0.655	37.7

Table 5: Résultats pour le clustering du prompt, selon la méthode

Le tableau 5 présente les résultats des approches basées sur le machine learning et sur la similarité cosinus. Le taux de bonne classification est de 0.655 pour l'approche basée sur la similarité cosinus, contre 0.609 pour l'approche machine learning. Cependant, l'approche machine learning est plus rapide pour traiter 1000 questions, avec un temps d'exécution de 18.4 secondes. Étant donné que la priorité est la qualité de la classification, nous choisissons l'approche basée sur la similarité cosinus. De plus, un temps d'exécution de 37.7 secondes pour 1000 questions correspond à 37.7 millisecondes par question, ce qui reste tout à fait satisfaisant.

Voyons maintenant si l'ajout de clusters pour la recherche sémantique améliore les résultats du RAG.

	RAG	RAG w/clustering
Similarité cosinus	0.542	0.532
ROUGE-1	0.267	0.255

Table 6: Scores de similarité entre le RAG et le RAG avec clustrering, avec les données SM

Le tableau 6 montre les scores de similarité pour le RAG avec et sans clustering. Bien que les résultats soient très proches, le RAG classique obtient des performances légèrement supérieures pour les deux métriques. L'ajout de clusters pour la recherche sémantique n'a pas permis d'améliorer les résultats du RAG. Cependant, l'utilisation des clusters mérite d'être explorée davantage. L'utilisation des clusters peut sans doute être optimisée afin d'apporter des résultats plus prometteurs.

7 | Conclusions

Nous voici arrivés au terme de notre étude, qui nous a permis de mieux comprendre comment adapter des LLM pour des domaines précis à coût réduit. Pour terminer, nous résumerons les principales conclusions de l'étude, avec une synthèse des résultats et une recherche des limites et pistes d'amélioration. Nous aborderons également les aspects de mon stage, en évaluant son impact environnemental et ses bénéfices personnels.

7.1 | Conclusions de l'étude

Commençons par les conclusions de l'étude en présentant un résumé des résultats à retenir, suivi des pistes d'amélioration à envisager.

7.1.1 | Synthèse des résultats

Pour confronter le RAG, le fine-tuning et de nouvelles techniques de spécialisation des LLM, nous avons mis en place plusieurs procédures d'évaluation basées sur deux jeux de données de questions-réponses. Cette étude montre que le modèle Mistral 7B a donné les meilleurs résultats sur nos données. Nous avons également observé que le RAG et le fine-tuning ne sont pas les méthodes les plus efficaces pour la spécialisation des LLM. En effet, l'utilisation de techniques combinant le RAG et le fine-tuning, comme le RAFT, a produit de meilleurs résultats que ces deux approches seules. Concernant notre contribution, nous avons constaté que HDBSCAN est la méthode de clustering la plus efficace pour notre expérience. Même si le clustering n'a pas amélioré les performances du RAG, il reste une option intéressante à explorer davantage.

7.1.2 | Limites et perspectives d'amélioration

Bien que cette étude fournisse des résultats pertinents pour le développement de LLM spécialisés, il est important de reconnaître certaines limites notables. Tout d'abord, concernant les données. Nous avons choisi d'utiliser des paires de questions-réponses pour l'évaluation, or un utilisateur ne formule pas forcément sa demande sous forme de questions. De plus, les réponses générées à partir d'un modèle à paramètre réduit peuvent manquer de précision. Aussi, nous avons principalement traité des données en anglais, alors que

nous devons être en mesure de travailler avec plusieurs autres langues. C'est pour cela qu'EasyVista retravaille sur un jeu de données plus complet, notamment en utilisant GPT-4, un modèle plus puissant, et en générant des simulations de prompts plus variées et de différentes langues. Un autre problème réside dans l'évaluation de nos générations. En effet, lorsque nous évaluons les textes, la réponse de référence peut contenir moins de détails que la réponse générée, ce qui peut entraîner des résultats plus faibles pour une réponse correcte. Aujourd'hui encore, l'évaluation des réponses des modèles reste difficile.

L'IA générative reste une technologie jeune où les avancées progressent à une vitesse fulgurante. Régulièrement, de nouvelles études tentent d'apporter leur pierre à l'édifice, et cela concerne également la spécialisation de LLM. Aujourd'hui, les recherches se concentrent essentiellement sur l'amélioration du RAG et du fine-tuning. Par exemple, nous pouvons noter deux techniques récentes pour le RAG : le Reranking [38] et GraphRAG [9]. Le Reranking est une étape supplémentaire qui permet d'améliorer la pertinence des documents récupérés en les réordonnant selon leur importance. GraphRAG utilise les grands graphes de connaissances pour unifier les informations pertinentes de plusieurs documents connectés.

Ces avancées ouvrent des portes à explorer pour EasyVista, qui, comme nous l'avons évoqué, a déjà commencé des recherches sur les meilleurs paramètres pour le RAG. Il est donc nécessaire de se tenir informé des progrès en cours. La seule certitude que nous pouvons avoir est que la vérité d'aujourd'hui sera remplacée par les découvertes de demain.

7.2 | Conclusions du stage

L'heure est au bilan général du stage. Nous commencerons par évaluer l'impact du projet sur l'environnement. Ensuite, nous ferons le point sur le stage dans son ensemble, en nous focalisant sur les bénéfices personnels et les compétences acquises au cours de cette expérience.

7.2.1 | Impact environnemental

Pour évaluer l'impact environnemental de ce stage, nous avons pris en compte deux types de dépenses : celles liées au transport et celles liées au numérique. Concernant le transport, nous avons compté le nombre total de trajets effectués au cours des 6 mois de stage, en fonction du moyen de transport utilisé. Pour l'impact numérique, nous avons noté le temps d'utilisation du PC et le temps d'utilisation d'un GPU de 16 Go (NVIDIA T4). Regardons les résultats obtenus.

Au cours de ce stage, 334 trajets de 3.6 km ont été réalisés. Parmi eux, 222 ont été réalisés en voiture (GPL thermique), 90 à vélo électrique, et 22 en covoiturage. Le gouvernement français et l'ADEME (Agence De l'Environnement et de la Maîtrise de l'Énergie) ont développé un outil pour calculer la consommation en équivalent CO2 (CO2eq) selon le mode de transport [4]. Pour un trajet de 3.6 km, la consommation est de 0.78 kg CO2eq en voiture, de 0.04 kg CO2eq à vélo électrique, et de 0.39 kg CO2eq en covoiturage. Cela correspond à un total de 173.16 kg CO2eq pour les trajets en voiture, 3.6 kg CO2eq pour ceux à vélo électrique, et 8.58 kg CO2eq pour le covoiturage, soit un total de 185.34 kg CO2eq lié aux déplacements durant ce stage.

En ce qui concerne la consommation liée au numérique, nous avons compté un temps d'utilisation de 784 heures pour le PC, et de 106 heures pour le GPU. Un PC portable branché consomme environ 80 Wh [8], ce qui correspond à une consommation de 62.72 kWh pour l'ordinateur. Le GPU, de son côté, consomme 70 Wh [27], soit 7.42 kWh. Ainsi, la consommation totale durant ce stage liée au numérique s'élève à 70.14 kWh. En France, 1 kWh équivaut à environ 0.1 kgCO2eq [7], ce qui représente un total de 7.014 kg CO2eq.

Ce stage aura représenté une consommation énergétique totale de 192.354 kg CO2eq. Ces résultats sont fortement impactés par les transports, et plus particulièrement par les déplacements en voiture. Une solution pour réduire drastiquement ces résultats serait d'oublier la voiture et de choisir des modes de transport plus sains. Si l'ensemble des trajets en voiture avait été remplacé par des trajets à vélo électrique, la consommation aurait été réduite par 14, passant de 185.34 kg CO2eq à 13.36 kg CO2eq. De son côté, EasyVista a mis en place une activité ludique pour prendre conscience de notre consommation énergétique liée à nos activités quotidiennes, divisée en cinq catégories : transports, logement, alimentation, biens et services. Grâce à une simulation de notre consommation quotidienne, l'objectif était de prendre des décisions collectives et individuelles pour réduire notre empreinte carbone et atteindre les 2 tonnes de CO2eq par habitant d'ici 2050, et ainsi respecter les engagements de l'Accord de Paris. Cet atelier m'a permis de réaliser l'impact de mes déplacements sur mon empreinte carbone. Si aujourd'hui je souhaite agir pour l'environnement, je dois alors limiter mes trajets en véhicule thermique et favoriser davantage des modes de transport plus écologiques.

7.2.2 | Bilan personnel

Mon stage chez EasyVista a été une première expérience en entreprise enrichissante. J'ai eu l'occasion de mettre en pratique mes connaissances acquises au cours de mes études dans un environnement professionnel, tout en découvrant de nouvelles facettes du métier, en particulier dans les domaines de l'IA générative et du NLP. J'ai eu l'opportunité de travailler aux côtés de professionnels compétents qui ont contribué à enrichir mes connaissances en IA.

Ce stage a renforcé mon envie de poursuivre une carrière en tant que Data Scientist. L'aventure chez EasyVista n'était que le début de mon parcours. À présent, mon objectif est de vivre de nouvelles expériences similaires, tout en continuant à explorer le vaste monde de la statistique et des sciences des données.

References

[1] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, William El Sayed. "Mistral 7B". In: (2023). URL: https://arxiv.org/pdf/2310.06825.

[2] *Applications that can reason. Powered by LangChain.* URL: https://www.langchain.com/.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. "Attention Is All You Need". In: (2017). URL: https://arxiv.org/pdf/1706.03762.

[4] *Calculer les émissions de carbone de vos trajets.* URL: https://agirpourlatransition.ademe.fr/particuliers/bureau/calculer-emissions-carbone-trajets.

[5] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: (2004). URL: https://aclanthology.org/W04-1013.pdf.

[6] *Chroma.* URL: https://docs.trychroma.com/.

[7] *Combien de CO2 dégage un 1 kWh électrique ?* URL: https://www.greenit.fr/2009/04/24/combien-de-co2-degage-un-1-kwh-electrique/#:~:text=En%20France%2C%20un%20kWh%20%C3%A9lectrique,CO2%20eq.%20%2F%20kWh%20%C3%A9l..

[8] *Comment évaluer la consommation électrique de mon PC ?* URL: https://www.happ-e.fr/actualites/les-eco-gestes/quelle-est-consommation-electrique-dun-pc#:~:text=La%20consommation%20%C3%A9lectrique%20d'un%20PC%20de%20bureau%20%C2%AB%20standard%20%C2%BB,90%20%E2%82%AC*%20sur%20votre%20facture..

[9] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Jonathan Larson. "From Local to Global: A Graph RAG Approach to Query-Focused Summarization". In: (2024). URL: https://arxiv.org/pdf/2404.16130.

[10] *Easily finetune train LLMs Get faster with unsloth.* URL: https://unsloth.ai/.

[11] *GPT4 et ses 1.8 billion de paramètres - Comment atteindre l'ultra pertinence ?* URL: https://ai-side.com/fr/news/nouveau-leak-gpt-4-decryptage.

[12] *Hugging Face.* URL: https://huggingface.co/.

[13] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, Song Han. "AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration". In: (2023). URL: https://arxiv.org/pdf/2306.00978.

[14] Juergen Schmidhuber. "Deep Learning in Neural Networks: An Overview". In: (2014). URL: https://arxiv.org/pdf/1404.7828.

[15] *K-means : Focus sur cet algorithme de Clustering et Machine Learning.* URL: https://datascientest.com/algorithme-des-k-means.

[16] Kai Lv, Yuqing Yang, Tengxiao Liu, Qinghui Gao, Qipeng Guo, Xipeng Qiu. "Full Parameter Fine-tuning for Large Language Models with Limited Resources". In: (2023). URL: https://arxiv.org/pdf/2306.09782.

[17] Kevin Wu, Eric Wu, James Zou. "How faithful are RAG models? Quantifying the tug-of-war between RAG and LLMs' internal prior". In: (2024). URL: https://arxiv.org/html/2404.10198v1.

[18] Leland McInnes, John Healy, James Melville. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: (2018). URL: https://arxiv.org/pdf/1802.03426.

[19] Lizhou Fan, Lingyao Li, Zihui Ma, Sanggyu Lee, Huizi Yu, Libby Hemphill. "A Bibliometric Review of Large Language Models Research from 2017 to 2023". In: (2023). URL: https://arxiv.org/pdf/2304.02020.

[20] Louis SANT'ANNA. "La Similarité Cosinus en IA/NLP". In: (2023). URL: https://medium.com/@santannalouis208/la-similarit%C3%A9-cosinus-en-ia-nlp-d554d3b14efa.

[21] Maarten Grootendorst. "Topic Modeling with BERT". In: (2020). URL: https://towardsdatascience.com/topic-modeling-with-bert-779f7db187e6.

[22] Mathav Raj J, Kushala VM, Harikrishna Warrier, Yogesh Gupta. "Fine tuning llms for enterprise: practical guidelines and recommendations". In: (2024). URL: https://arxiv.org/pdf/2404.10779.

[23] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, Hervé Jégou. "The Faiss library". In: (2024). URL: https://arxiv.org/pdf/2401.08281.

[24] Md Farhadur Rahman, Weimo Liu, Saad Bin Suhaim, Saravanan Thirumuruganathan, Nan Zhang, Gautam Das. "HDBSCAN: Density based Clustering over Location Based Services". In: (2016). URL: https://arxiv.org/pdf/1602.03730.

[25] *Méthode des k plus proches voisins*. URL: https://fr.wikipedia.org/wiki/M%C3%A9thode_des_k_plus_proches_voisins.

[26] Nils Reimers, Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: (2019). URL: https://arxiv.org/pdf/1908.10084.

[27] *NVIDIA T4*. URL: https://www.nvidia.com/fr-fr/data-center/tesla-t4/.

[28] Peter J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: (1987). URL: https://www.sciencedirect.com/science/article/pii/0377042787901257?via%3Dihub.

[29] *Pycaret, low-code machine learning*. URL: https://pycaret.org/.

[30] Roberto Gozalo-Brizuela, Eduardo C. Garrido-Merchán. "A survey of Generative AI Applications". In: (2023). URL: https://arxiv.org/pdf/2306.02781.

[31] *SentenceTransformers Documentation*. URL: https://www.sbert.net/index.html.

[32] Tianjun Zhang, Shishir G. Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, Joseph E. Gonzalez. "RAFT: Adapting Language Model to Domain Specific RAG". In: (2024). URL: https://arxiv.org/pdf/2403.10131.

[33] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, Luke Zettlemoyer. "QLoRA: Efficient Finetuning of Quantized LLMs". In: (2023). URL: https://arxiv.org/pdf/2305.14314.

[34] *Tokenization in Machine Learning Explained*. URL: https://vaclavkosar.com/ml/Tokenization-in-Machine-Learning-Explained.

[35] *Trainer - Hugging Face*. URL: https://huggingface.co/docs/transformers/v4.15.0/en/main_classes/trainer#transformers.TrainingArguments.

[36] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, Ji-Rong Wen. "A Survey of Large Language Models". In: (2023). URL: https://arxiv.org/pdf/2303.18223.

[37] *wiki EasyVista*. URL: https://wiki.easyvista.com/xwiki/bin/view/Documentation/.

[38] Xiaohua Wang, Zhenghua Wang, Xuan Gao, Feiran Zhang, Yixin Wu, Zhibo Xu, Tianyuan Shi, Zhengyuan Wang, Shizheng Li, Qi Qian, Ruicheng Yin, Changze Lv, Xiaoqing Zheng, Xuanjing Huang. "Searching for Best Practices in Retrieval-Augmented Generations". In: (2024). URL: https://arxiv.org/pdf/2407.01219.

[39] *Yahoo! Answers Topic Classification*. URL: https://www.kaggle.com/datasets/bhavikardeshna/yahoo-email-classification.

[40] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, Haofen Wang. "Retrieval-Augmented Generation for Large Language Models: A Survey". In: (2023). URL: https://arxiv.org/pdf/2312.10997.