

Remerciement

C'est avec une sincère reconnaissance que je consacre cette section pour exprimer ma profonde gratitude envers toutes les personnes qui ont été d'un soutien indispensable pour la réussite de ce projet.

En cette occasion spéciale, nous tenons à exprimer notre gratitude envers la coordonnatrice du Master IARV, Madame **RAJAA TOUAHNI**, pour la qualité de la formation qu'elle nous a apportée, ainsi que pour son soutien et son accompagnement tout au long de notre parcours.

Nos sincères remerciements vont également à notre encadrant interne, M. **KAICER Mohammed**, pour son précieux guidage tout au long de ce travail. Nous tenons également à remercier les membres du jury pour avoir accepté d'examiner et d'évaluer notre travail.

Nous exprimons également notre gratitude envers M. **Mehdi Nassiri**, notre encadrant externe, pour son accueil chaleureux, son encadrement efficace et son accompagnement tout au long de cette expérience professionnelle enrichissante.

Un merci spécial est adressé à M. **Youssef ERMILI** et M. **Mohamed Amine KHAMMOUR** pour leur dévouement et leur soutien précieux tout au long de cette aventure. Leur expertise et leur disponibilité ont été des atouts précieux, nous permettant d'avancer avec confiance et détermination.

Avec beaucoup de respect, nous exprimons notre grande gratitude au corps professoral du **Master intelligence artificielle et réalité virtuelle** pour avoir porté un vif intérêt à notre formation, et pour nous avoir consacré le plus clair de leur temps, de leur attention et de leur énergie dans un cadre agréable de complicité et de respect.

Je tiens à exprimer ma plus profonde gratitude envers ma famille. À ma mère et mon père, pour leur amour inconditionnel, leur soutien constant et leurs sacrifices qui m'ont permis d'arriver où je suis aujourd'hui. À mon frère et ma sœur, pour leur encouragement, leur complicité et leur présence à mes côtés tout au long de ce parcours. Votre soutien a été une source inépuisable de motivation et de réconfort.

Nous tenons également à exprimer notre reconnaissance envers toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce projet. Leur implication et leur soutien ont été essentiels, et nous les remercions chaleureusement pour leur précieuse collaboration.

Résumé

Ce rapport constitue une synthèse de notre stage de fin d'études réalisé au sein de l'entreprise PEAQOCK à Casablanca, dans le cadre de notre diplôme de Master en Intelligence Artificielle et Réalité Virtuelle (IARV) à la Faculté des Sciences de l'Université Ibn Tofail (UIT) de Kénitra.

Le projet visait à développer un modèle capable de comprendre le langage naturel et de le traduire avec précision en requêtes SQL. Intitulé « Fine-tuning a Model for Translating Natural Language to SQL Queries », ce projet avait pour objectif d'automatiser la conversion des descriptions textuelles en commandes SQL exécutables, en utilisant un schéma fourni pour générer des requêtes SQL valides.

La capacité à traduire automatiquement le langage naturel en requêtes SQL est cruciale pour les systèmes utilisant des bases de données.

Mon projet chez PEAQOCK Financials s'inscrit dans cette démarche d'innovation technologique, visant à améliorer l'accessibilité et l'efficacité de l'interaction avec les bases de données.

En plus de ce projet principal, durant mon stage, j'ai également travaillé sur d'autres tâches incluant la gestion des bases de données et le développement backend.

Pour réaliser nos objectifs, nous avons débuté par une analyse approfondie des besoins et des spécifications fonctionnelles, suivie par une phase de collecte de données et d'étude des méthodes de fine-tuning des modèles de langage de grande taille (LLM). Ensuite, nous avons commencé la phase de fine-tuning, tout en adoptant la méthodologie agile Scrum, ce qui a facilité une exécution efficace et adaptable du projet.

La structure de ce document sera la suivante :

Le premier chapitre présentera le contexte général du projet, comprenant une introduction à l'entreprise PEAQOCK et une brève description de la tâche assignée. Le deuxième chapitre abordera le contexte métier du projet, en mettant en lumière le domaine des LLM et des modèles de conversion de texte en code, spécifiquement TEXT-TO-SQL. Le troisième chapitre sera dédié à la description des différentes technologies adoptées pendant la réalisation du projet. Enfin, le quatrième chapitre exposera le processus et les différentes étapes de la réalisation du projet, détaillera les résultats obtenus, l'état d'avancement du projet et les suggestions d'améliorations.

Mots clés : PEAQOCK, TEXT-TO-CODE, TEXT-TO-SQL, base de données, SCRUM.

Abstract

This report summarizes our final-year internship at PEAQOCK in Casablanca, part of our master's degree in Artificial Intelligence and Virtual Reality (IARV) at the Faculty of Sciences, Ibn Tofail University (UIT) in Kenitra.

The project aimed to develop a model capable of understanding natural language and accurately translating it into SQL queries. Titled "Fine-tuning a Model for Translating Natural Language to SQL Queries," this project sought to automate the conversion of textual descriptions into executable SQL commands using a provided schema to generate valid SQL queries.

The ability to automatically translate natural language into SQL queries is crucial for systems that use databases. My project at PEAQOCK Financials is part of this technological innovation effort, aiming to improve the accessibility and efficiency of database interactions.

In addition to this main project, during my internship, I also worked on other tasks including database management and backend development.

To achieve our objectives, we started with an in-depth analysis of the needs and functional specifications, followed by a data collection phase and study of fine-tuning methods for large language models (LLM). Then, we began the fine-tuning phase, all while adopting the agile Scrum methodology, which facilitated efficient and adaptable project execution.

The structure of this document will be as follows:

The first chapter will present the general context of the project, including an introduction to PEAQOCK and a brief description of the assigned task. The second chapter will discuss the business context of the project, highlighting the field of large language models (LLM) and text-to-code models, specifically TEXT-TO-SQL. The third chapter will be dedicated to describing the various technologies adopted during the project. Finally, the fourth chapter will describe the process and the various stages of the project's realization, detail the results obtained, the project's progress, and suggestions for improvements.

Keywords: PEAQOCK, TEXT-TO-CODE, TEXT-TO-SQL, database, SCRUM.

Liste des abréviations

Abréviation	Signification
NL	Natural language
LLM	Large language model
IA	Intelligence Artificielle (Artificial Intelligence)
ML	Machine Learning (Apprentissage automatique)
DL	Deep learning
SQL	Structured Query Language (Langage de requête structuré)
IDE	Integrated Development Environment (Environnement de développement intégré)
DBMS	Database Management System (Système de gestion de base de données)
API	Application Programming Interface (Interface de programmation d'applications)
ETL	Extract, Transform, Load (Extraire, Transformer, Charger)
Dataset	Ensemble de Données
Codex	Modèle de Génération de Code d'OpenAI
T5	Transformateur de Texte à Texte
Schema	Les instructions pour création d'une base de données

Table des matières

Remerciement.....	2
Résumé	3
Abstract.....	4
Liste des abréviations	5
Table des matières	6
Chapitre 1. Introduction générale.....	8
Introduction.....	9
1.1. Présentation de l'organisme d'accueil	9
1.1.1. Organigramme du PEAQOCK	10
1.1.2. Centres de Services	11
1.1.3. Valeurs Fondamentales	12
1.1.4. Solutions de Peaqock.....	12
1.1.5. Partenaires.....	16
1.1.6. Clients	17
1.1.7. Prix internationaux.....	18
1.2. Travail demandé.....	18
1.3. Présentation de l'équipe.....	20
1.4. Tâches attribuées.....	21
1.5. Contexte métier	21
1.6. Description de contenu.....	21
Conclusion	22
Chapitre 2. L'état de l'art des LLM	23
Introduction.....	24
2.1. Les large language models : LLMs.....	25
2.1.1. Définition des LLMs.....	25
2.1.2. Types des LLMs.....	25
2.1.3. Les models Text-to-code et Text-to-Sql.....	27
2.1.4. L'état de l'art.....	30
Conclusion	43
Chapitre 3. Outils et Technologies utilisés	44
Introduction.....	45
3.1. Language, Frameworks et bibliothèques	45
3.1.1. Language de programmations	45

3.1.2. Plateformes :	47
3.1.3. Bibliothèques	49
3.2. Systèmes de gestion de base de données	52
3.2.1. Postgres	52
3.2.2. MongoDB	52
3.2.3. Hugging-Face datasets	53
3.3. Outils de communications et de réunions	54
3.3.1. Google Meet	54
3.3.2. Google chat	55
3.4. Outils d'industrialisation	55
3.4.1. Docker	55
3.4.2. Git	57
3.4.3. GitHub	58
3.4.4. PyCharm	58
Conclusion	60
Chapitre 4. Réalisation et concrétisation	61
Introduction	62
4.1. Démarche suivie	62
4.2. Collection des données pour fine-tuning	62
4.3. Choix du model	64
4.4. Choix de la méthode du fine-tuning	65
4.5. Code du fine-tuning	67
4.6. Evaluation des résultats de performance et rapidité	71
4.7. Le model en action dans OneGate	74
Conclusion	75
Conclusion : Lesson, Défis et Perspectives d'avenir	76
Liste des figures	79
Liste des tableaux	81
Références	82

Chapitre 1. Introduction générale

Introduction

Dans ce chapitre introductif, nous présenterons tout d'abord l'organisme d'accueil PEAQOCK, en citant ses clients, ses valeurs fondamentales et ses solutions innovantes. Ensuite, nous procéderons à une présentation générale du projet et nous aborderons l'organisation de l'équipe de travail, les tâches qui nous ont été attribuées dans le contexte métier, ainsi que les technologies utilisées.

1.1. Présentation de l'organisme d'accueil



Figure 1.1: logo PEAQOCK

PEAQOCK est une entreprise dynamique spécialisée dans les solutions innovantes en matière de science des données et d'intelligence artificielle. Elle fournit une gamme de solutions pour les industries bancaire, d'assurance et de commerce. Ses plateformes ont été conçues pour être facilement manipulables, adaptables et conformes aux dernières normes technologiques, les rendant ainsi parfaitement adaptées à l'ère de la transformation digitale. Son objectif est de faire gagner du temps et de l'argent à ses clients grâce à des rapports, une gestion des données et une sécurisation des données. Elle est dévouée à fournir des services innovants en utilisant les technologies les plus récentes telles que l'intelligence artificielle et la blockchain, pour permettre aux entreprises de rester en avance sur la concurrence. Avec des bureaux à Casablanca et Paris, elle sert une large gamme de clients à travers le monde et les aide à rationaliser leurs processus d'affaires grâce à l'utilisation d'outils numériques avancés.

Création	2014
Fondateur & CEO	Mohammed BENKHALED
Forme juridique	Société à Responsabilité Limitée
Siège social	Immeuble Zénith II, 1er étage, Sidi Maarouf, Casablanca
Activité	Services et conseils en informatique
Site web	www.peaqock.com
Chiffre d'affaires	De 5,000,000 à 10,000,000 dh (2023)
Capital	400,000.00 dh

Table 1: fiche d'identité de PEAQOCK

PEAQOCK a été fondée dans le but de proposer une alternative innovante au schéma classique du monde des nouvelles technologies en accompagnant les grandes structures, opérant au Maroc, en Afrique, en Europe et en Asie, dans la réussite de leurs projets. Elle essaie continuellement de mettre sur le marché des produits nouveaux et innovantes qui répondent aux besoins des clients.

1.1.1. Organigramme du PEAQOCK

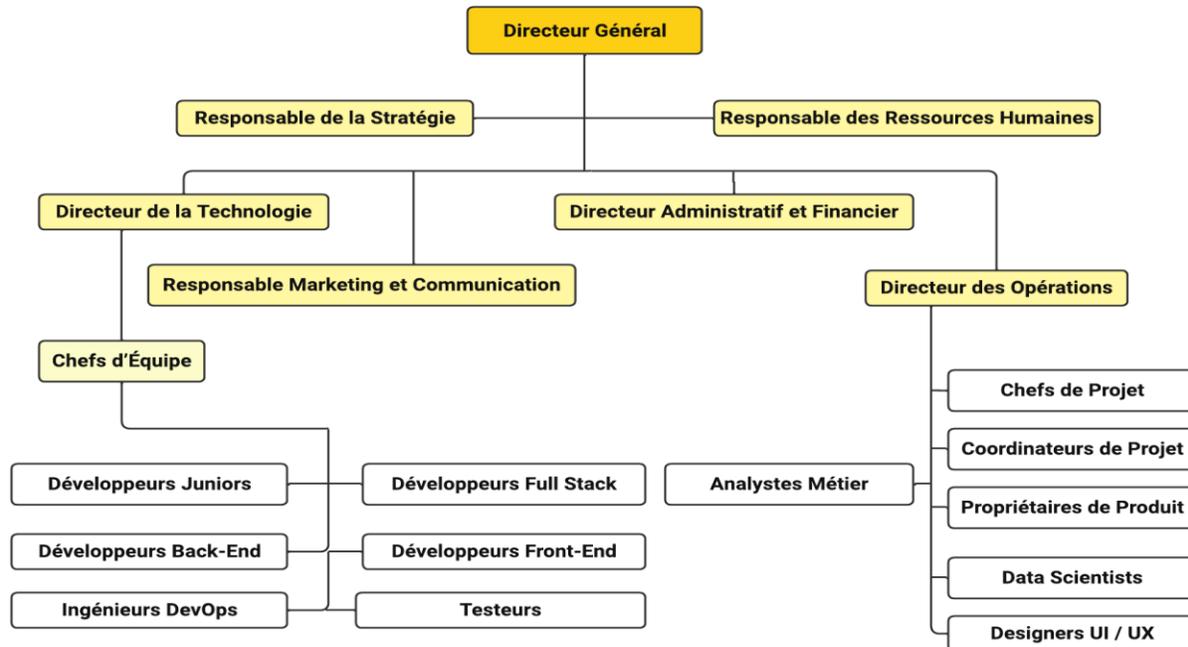


Figure 1.2 : l'organigramme de l'entreprise PEAQOCK

L'entreprise PEAQOCK est composée de plusieurs départements, chacun jouant un rôle bien précis et déterminé.

- ◆ Les chefs d'équipe (Team Leaders) répartissent les tâches au sein de l'équipe, suivent leur avancement et participent aux réunions de coordination avec le reste de l'entreprise et les réunions avec les clients afin de mettre l'accent sur les besoins des clients.
- ◆ Les directeurs des projets (Project Managers) ont la responsabilité de la planification, de l'approvisionnement et de l'exécution des projets dans l'entreprise. Ils participent rarement aux activités de production du résultat final.
- ◆ Les coordinateurs de projets (Project Coordinators) sont une partie importante de l'équipe des projets, s'assurant que les projets sont terminés dans les délais prévus et dans les limites du budget.
- ◆ Le propriétaire de produit (Product Owner) qui est chargé de satisfaire les besoins des clients en assurant la livraison d'un produit ou service de qualité, et il est le premier responsable de la conception de produit.
- ◆ Les scientifiques des données (Data Scientists) sont chargés de la gestion, de l'analyse et de l'exploitation des données au sein de l'entreprise.
- ◆ Les designers UI/UX (Designer UI/UX) sont responsables de la conception générale de l'interface utilisateur et s'intéressent à l'expérience utilisateur.

- ◆ Les ingénieurs DevOps assurent le déploiement, l'automatisation, la maintenance et la gestion de l'infrastructure logicielle dans l'entreprise.
- ◆ Les développeurs full stack sont responsables du développement tant côté client (frontend) que côté serveur (backend) d'une application.
- ◆ Les développeurs frontend se concentrent sur le développement de l'interface utilisateur d'une application web ou mobile.
- ◆ Les développeurs backend se concentrent sur la mise en place des fonctionnalités et de la logique métier côté serveur d'une application.
- ◆ Les juniors sont des membres récents de l'équipe, souvent en début de carrière, qui sont en phase d'apprentissage et de développement de leurs compétences professionnelles.
- ◆ Les testeurs sont chargés de la validation et de la vérification des fonctionnalités logicielles afin d'assurer la qualité et la fiabilité du produit final.

1.1.2. Centres de Services

Peaqqock Financials, une entreprise spécialisée dans les Services Financiers, se distingue par son expertise dans les domaines de la Data et des Paiements. En comprenant les implications des données et l'impact de la Fintech sur les marchés traditionnels, Peaqqock Financials offre des conseils précieux aux banques, régulateurs et assurances pour saisir les opportunités et relever les défis de l'ère numérique. Leur engagement solide consiste à offrir des services innovants, exploitant les technologies de pointe telles que l'intelligence artificielle, la blockchain et l'apprentissage automatique. Ainsi, ils permettent aux entreprises de maintenir une longueur d'avance sur leurs concurrents, tout en garantissant leur succès dans un monde en constante évolution.

Fidèle à sa vision, **Peaqqock** fournit une gamme complète de solutions sur mesure pour les secteurs bancaire, assurantiel et commercial. Leurs services comprennent :

- **Conseil stratégique :**

Lors de chaque rendez-vous d'exploration avec leurs prospects, ils essaient de comprendre en profondeur le contexte culturel et les grands objectifs qui animent leur démarche quotidienne. Ils offrent aux clients le meilleur de leur expertise pour comprendre les avantages et les limites de leurs modèles économiques et comment ils peuvent innover dans leurs processus.

- **Data Management :**

Aider les clients à choisir leur architecture de données (big data ou data management traditionnel) et étudier les différentes sources de données (outils bureautiques, réseaux sociaux, etc...)

- **Data Science :**

Au-delà de l'aspect nettoyage et structuration des données, Peaqqock Financials accompagne ses clients pour construire les meilleurs modèles basés sur l'IA et/ou le machine learning pour améliorer la rentabilité, faire une étude de profilage pour mieux cibler les clients, augmenter l'efficacité opérationnelle, prédire les ventes futures, etc.

- **Visualisation des données :**

Fournir les meilleurs outils pour visualiser les KPI afin que les décideurs aient accès aux données de manière optimisée avec tous les scénarios possibles.

Leurs plateformes conviviales sont flexibles et parfaitement adaptées aux besoins spécifiques de chaque client. En utilisant les dernières normes et technologies de communication, Peaqqock Financials assure une expérience fluide et hautement performante, adaptée à l'ère de la transformation numérique.

Ainsi, les clients de Peacock Financials bénéficient de solutions technologiques qui non seulement répondent à leurs besoins actuels, mais également leur permettent de se préparer efficacement à l'avenir.

1.1.3. Valeurs Fondamentales

Chez PEAQOCK, nous sommes guidés par un ensemble de valeurs fondamentales qui définissent notre culture d'entreprise et guident nos actions au quotidien.

- ◆ **Innovation** : Chez PEAQOCK, nous croyons fermement à l'importance de l'innovation pour rester compétitif sur le marché. Nous investissons dans la recherche et le développement pour générer de nouvelles idées, produits et processus. Cette culture de l'innovation nous permet de répondre aux besoins changeants de nos clients et de rester en avance sur nos concurrents.
- ◆ **Excellence** : Nous nous efforçons constamment d'atteindre l'excellence dans tout ce que nous faisons. Cela signifie que nous sommes toujours à la recherche de moyens d'améliorer nos services, nos produits et nos processus. Nous visons le plus haut niveau de qualité et de performance afin de garantir la satisfaction de nos clients et de maintenir notre réputation d'excellence.
- ◆ **Ambition** : Chez PEAQOCK, nous sommes guidés par une ambition sans faille. Nous sommes déterminés à atteindre le succès, que ce soit pour nos clients, nos employés ou notre entreprise dans son ensemble. Nous sommes prêts à relever les défis qui se présentent sur notre chemin et à travailler dur pour les surmonter.
- ◆ **Éthique** : L'éthique est au cœur de tout ce que nous faisons chez PEAQOCK. Nous croyons en l'importance de l'intégrité, de la transparence et de l'honnêteté dans nos interactions avec nos clients, nos employés et nos partenaires commerciaux. Nous sommes fiers de mener nos activités de manière éthique et responsable, en respectant les normes les plus élevées de conduite professionnelle.

1.1.4. Solutions de Peacock

Peacock Financials propose une gamme de produits et de services axés sur la gestion efficace des données, la visualisation convaincante et l'optimisation des processus internes. Ces produits sont spécialement conçus pour répondre aux besoins des secteurs de la banque, de l'assurance et du commerce.

1.1.4.1. PEAQOCK OneGate



Figure 1.3: logo OneGate

OneGate est une plateforme conçue pour améliorer l'interopérabilité des données entre diverses applications commerciales. Cette solution aide les **business owners** à bien comprendre leur Data sans avoir juste en utilisant du langage naturel est sans connaissance des langages de programmation et sans utilisation de SQL.

OneGate offre une diverse gamme de services :

Recherche Intelligente :

La recherche Intelligente est un module de recherche intelligent alimenté par l'IA offert par OneGate qui simplifie les recherches quotidiennes en convertissant des entrées textuelles ou vocales basiques en résultats efficaces, tout en exploitant les données exclusives de l'entreprise.

Consultance en IA :

La consultance en IA permet aux membres de l'organisation de synerger avec les données privées de l'entreprise et les responsabilités individuelles, fournissant un support personnalisé pour les collaborations de projet, la génération de rapports et le développement de la stratégie de l'entreprise.

Aperçu Commercial :

Le module est un atout crucial pour les organisations car il génère rapidement des aperçus et des prévisions de données, permettant d'obtenir des informations précieuses en quelques clics, plutôt que d'attendre des jours.

1.1.4.2. *PEAQOCK Fraude & Risk Management Solution*



Figure 1.4: logo fraud & risk management solution

Peacock Financials propose une solution de gestion des fraudes et des risques, connue sous le nom de Peacock Fraud & Risk Management Solution. Cette solution vous aide à protéger les actifs de vos clients et votre marque. Avec l'évolution des techniques de fraude, les attaques frauduleuses deviennent de plus en plus sophistiquées et leurs conséquences sont de plus en plus graves. Les institutions financières ont besoin d'une capacité d'identification précise de la fraude en temps réel et avant qu'elle ne se produise.

La solution de Peacock Financials utilise une combinaison d'algorithmes avancés basés sur l'IA et de règles métier expertes pour bloquer les transactions suspectes en se basant sur une analyse comportementale en temps réel. Grâce à cette approche proactive, ils aident les institutions financières à prévenir la fraude et à minimiser les risques associés.

1.1.4.3. AML



Figure 1.5: logo AML

Peacock Financials propose une solution de prévention des crimes financiers, connue sous le nom de Peacock Financials Crime Prevention. Cette solution aide les banques à prévenir l'utilisation de leur entreprise pour le blanchiment des produits issus de la criminalité financière ou pour des pratiques corrompues. Avec la transition rapide des clients des services bancaires traditionnels vers les services bancaires numériques, de plus en plus de canaux sont ajoutés pour maintenir une expérience client cohérente et positive. Cependant, les banques font face à des difficultés pour contrôler tous ces nouveaux points d'accès, ce qui les expose à un risque accru de criminalité financière et de préjudice à leur réputation.

La solution de Peacock Financials offre une approche complète pour identifier et prévenir les activités criminelles et les pratiques corrompues. En utilisant des techniques avancées d'analyse des données et d'apprentissage automatique, ils aident les banques à détecter les comportements suspects, à bloquer les transactions illicites et à maintenir l'intégrité de leur entreprise.

1.1.4.4. PFM



Figure 1.6: logo PFM

Le logiciel de gestion financière personnelle (PFM) aide les utilisateurs à gérer leur argent, à catégoriser les transactions, à regrouper les comptes de plusieurs institutions en une seule vue, à visualiser les tendances de dépenses, les budgets et la valeur nette. Il permet de comprendre sa situation financière afin de tirer le meilleur parti de ses actifs au quotidien et de planifier pour l'avenir.

Grâce au PFM, vous pouvez avoir une vue d'ensemble de vos finances, en ayant la possibilité de suivre vos revenus et vos dépenses, de définir des objectifs financiers, d'établir des budgets et de surveiller votre valeur nette. Le logiciel facilite également la visualisation de vos tendances de dépenses, ce qui vous permet de mieux comprendre vos habitudes de consommation et d'ajuster vos dépenses en conséquence.

En regroupant vos comptes bancaires, vos cartes de crédit et vos autres comptes financiers dans une seule interface, le PFM offre une vue consolidée de vos finances, ce qui facilite la gestion de votre argent. Vous pouvez suivre vos transactions, analyser vos dépenses par catégorie, recevoir

des alertes sur les dépenses excessives et prendre des décisions éclairées sur la façon d'allouer votre argent de manière plus efficace.

Le PFM est un outil précieux pour la gestion financière personnelle, vous permettant de mieux contrôler vos finances, d'économiser de l'argent et de prendre des décisions éclairées en matière de planification financière. Avec une interface conviviale et des fonctionnalités avancées, il simplifie le suivi et la gestion de vos finances personnelles.

1.1.4.5. *Tableau de bord dynamique*



Figure 1.7: *logo tableau de bord dynamique*

Conçu pour être entièrement configurable à 360°, ce tableau de bord dynamique est capable de traiter tous types de données provenant de différentes sources (bases de données, fichiers Excel, PDF, images, manuels...) afin d'exécuter des analyses en temps réel et de visualiser les indicateurs nécessaires.

Grâce à ce tableau de bord dynamique, vous pouvez agréger et analyser des données provenant de diverses sources, vous permettant ainsi d'avoir une vue complète de votre activité. Que ce soient des données structurées ou non structurées, notre solution est capable de les traiter et de les afficher de manière conviviale.

Vous pouvez configurer le tableau de bord selon vos besoins spécifiques, en choisissant les indicateurs clés que vous souhaitez suivre et en personnalisant l'affichage des données. Il vous offre une visualisation claire et intuitive des métriques importantes, vous permettant ainsi de prendre des décisions éclairées et d'identifier les tendances et les problèmes potentiels en temps réel.

Ce tableau de bord dynamique offre une interface conviviale et une expérience utilisateur optimale. Il est conçu pour être flexible et évolutif, s'adaptant à vos besoins changeants et à l'évolution de votre entreprise. Vous pouvez également accéder au tableau de bord depuis n'importe quel appareil, vous permettant ainsi d'avoir une visibilité sur votre activité où que vous soyez.

Avec ce tableau de bord dynamique, vous disposez d'un outil puissant pour l'analyse des données en temps réel et la visualisation des métriques essentielles à votre activité. Profitez de cette

fonctionnalité pour optimiser vos prises de décision et améliorer la performance de votre entreprise.

1.1.4.6. Regtech



Figure 1.8: logo RegTech

Cette solution Regtech offre des solutions complètes pour faciliter le partage d'informations à travers la structuration des échanges entre différents départements, entités et organisations, ainsi que la création de flux d'informations dynamiques. Elle permet de simplifier les processus de conformité réglementaire en automatisant la collecte, l'analyse et la gestion des données liées aux réglementations. Grâce à cette solution Regtech, les entreprises peuvent améliorer leur conformité, réduire les risques et optimiser l'efficacité de leurs opérations.

1.1.4.7. Funds



Figure 1.9: logo funds

Cette solution permet de créer des rapports riches et automatiques avec un design sophistiqué à partir de données provenant de différentes sources. Les données sont adaptées et visualisées, et les informations peuvent être partagées dans différents formats. Grâce à notre solution de gestion des fonds, les utilisateurs peuvent générer des rapports détaillés et personnalisés, obtenir une vue d'ensemble de leurs activités financières et partager facilement ces informations avec les parties concernées.

1.1.5. Partenaires

1.1.5.1. FIS



Figure 1.10: logo FIS

FIS (Fidelity National Information Services) est une société mondiale de services financiers et de technologie, spécialisée dans les solutions bancaires et les services de paiement. En tant que partenaire de Peacock, FIS collabore pour intégrer les produits et les solutions de Peacock dans son écosystème financier. Cette collaboration permet d'offrir aux clients de FIS des fonctionnalités avancées en matière de gestion des données, d'analyse et de prévention de la

fraude, renforçant ainsi l'offre globale de FIS en matière de services financiers. Grâce à ce partenariat, Peacock bénéficie de l'expertise de FIS dans le secteur financier et peut étendre sa portée auprès d'une clientèle plus large.

1.1.5.2. Atos



Figure 1.11: logo ATOS

Atos est le leader international du numérique sécurisé et décarboné. Le Groupe propose à ses clients une approche de bout en bout en leur offrant des solutions numériques de pointe, des services de conseil, ainsi que des solutions de sécurité digitale et de décarbonation.

Pionnier des services et produits de décarbonation, Atos s'engage à atteindre la neutralité carbone pour l'ensemble de son organisation ainsi que pour ses clients et partenaires. C'est tous ensemble que nous repousserons les limites de l'excellence scientifique et technologique afin de permettre au plus grand nombre, de vivre, travailler et progresser durablement et en toute confiance dans l'espace informationnel.

Atos est une SE (Société Européenne) cotée sur Euronext Paris et fait partie des indices CAC 40 ESG et Next 20.

En outre, Atos est désormais responsable de la commercialisation de la solution de détection de fraude en temps réel que Peacock développe pour **HPS**.

1.1.6. Clients

1.1.6.1. HPS



Figure 1.12: logo HPS

C'est une société innovante de solutions de paiement à la pointe de l'industrie internationale des paiements, fournissant des solutions de paiement pour les émetteurs, les acquéreurs, les processeurs de cartes, les organisations de vente indépendantes (ISO), les détaillants, ainsi que pour les commutateurs nationaux et régionaux dans le monde entier.

PowerCARD est la suite complète de solutions de **HPS** qui couvre l'ensemble de la chaîne de valeur des paiements en permettant des paiements innovants grâce à sa solution omnicanale qui permet le traitement de toutes les transactions provenant de tous les canaux initiés par tous les moyens de paiement.

Fondée en 1995, **HPS** a connu une croissance rapide pour devenir l'une des principales sociétés technologiques mondiales de cartes et de paiements, avec plus de 500 clients dans 95 pays, leur fournissant une technologie de paiement de pointe.

La mission de **HPS** depuis le début a été de fournir à ses clients des solutions technologiques innovantes qui transcendent l'ensemble de la chaîne de valeur des paiements, leur permettant d'innover rapidement et de mettre facilement et efficacement de nouveaux produits sur le marché grâce au logiciel **PowerCARD** de **HPS**, qui est de plus en plus considéré comme l'avenir de l'industrie des paiements. **HPS** dispose de bureaux dans les principaux centres d'affaires en Afrique, en Europe, au Moyen-Orient et en Asie, d'une équipe de direction de classe mondiale et d'une réputation éprouvée en tant que l'un des fournisseurs de technologie de paiement les plus puissants de l'industrie.

1.1.6.2. Autres clients

PEAQOCK compte autres clients provenant principalement des secteurs de la finance, de la banque et de l'assurance. Voici la liste :



Figure 1.13: liste des clients de PEAQOCK

1.1.7. Prix internationaux

Voici une liste des prix internationaux remportés par **PEAQOCK**, témoignant de notre engagement envers l'excellence et l'innovation dans le domaine financier :

- ◆ **Forbes** : Reconnu par l'un des magazines les plus prestigieux au monde deux années consécutives.
- ◆ **Label finance innovation** : Certifié innovant par Paris Europlace (*le centre financier de Paris, a créé un pôle de compétitivité internationale appelé "Finance Innovation", qui a certifié les entreprises les plus innovantes à forte valeur ajoutée dans le secteur financier*).

1.2. Travail demandé

Dans le cadre de ce projet de fin d'études, le travail demandé consistait à affiner un modèle de langage de grande taille (LLM) pour la tâche de conversion du langage naturel en requêtes SQL. Cette mission comprenait plusieurs étapes clés :

1. **Collecte de données** : Réunir un ensemble de données pertinent et diversifié contenant des exemples de requêtes en langage naturel et leurs équivalents en SQL. Cela impliquait la

recherche de bases de données existantes, ainsi que la création de nouvelles données spécifiques aux besoins du projet.

2. **Préparation des données** : Effectuer le nettoyage et le prétraitement des données collectées afin de les rendre utilisables pour l'entraînement du modèle. Cette étape incluait la normalisation du texte, la gestion des valeurs manquantes et la transformation des données en un format compatible avec les algorithmes d'apprentissage automatique.
3. **Choix de la méthode d'affinage** : Sélectionner les méthodes d'affinage les plus appropriées pour le modèle de langage choisi. Cela impliquait l'évaluation de différentes techniques d'entraînement, telles que le fine-tuning supervisé, le transfer learning et l'entraînement avec des techniques d'augmentation de données.
4. **Sélection du modèle à affiner** : Identifier et choisir le modèle de langage le plus adapté aux spécificités de la tâche. Cette étape nécessitait une analyse comparative des modèles disponibles, en tenant compte de critères tels que la performance, la taille, les ressources nécessaires et les contraintes de temps.
5. **Mise en œuvre et expérimentation** : Affiner le modèle sélectionné en utilisant les données préparées et la méthode choisie. Effectuer des tests et des évaluations pour mesurer la performance du modèle et ajuster les paramètres d'entraînement en fonction des résultats obtenus.
6. **Documentation et analyse des résultats** : Documenter toutes les étapes du processus, analyser les résultats obtenus et formuler des conclusions sur l'efficacité de l'approche suivie. Proposer des recommandations pour des améliorations futures et des pistes de recherche complémentaires.

Le travail réalisé dans ce projet vise à démontrer la faisabilité et l'efficacité de l'affinage d'un modèle de langage de grande taille pour la conversion de langage naturel en requêtes SQL, en contribuant ainsi à l'amélioration des interactions homme-machine dans le domaine de la gestion de bases de données.

1.3. Présentation de l'équipe

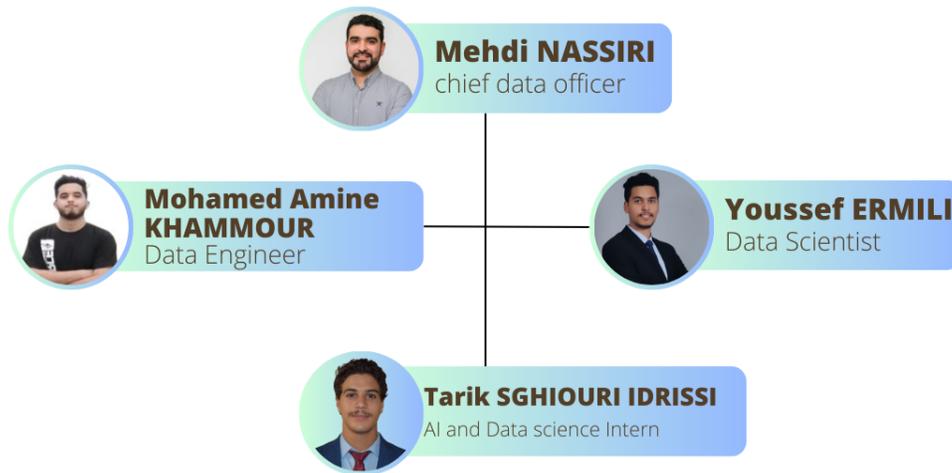


Figure 1.14: présentation d'équipe

Dans le cadre de ce projet, voici la composition de notre équipe :

- **Chief Data Officer (Mehdi NASSIRI):**
 1. Responsable de la supervision technique du projet.
 2. Contribution à la conception de l'architecture logicielle.
 3. Encadrement technique de l'équipe, notamment en ce qui concerne les technologies à utiliser et les bons pratiques.
- **Data Engineer (Mohamed Amine KHAMMOUR):**
 1. Supervision générale du stagiaire et support dans la compréhension des objectifs du projet.
 2. Guidance sur les aspects techniques et méthodologiques du traitement et de la préparation des données.
 3. Évaluation du travail réalisé et validation des livrables.
 4. Implémentation de certaines fonctionnalités liées à la manipulation et au prétraitement des données.
- **Data scientist (Youssef ERMILI) :**
 1. Suggestions et implémentation à propos des modèles.
 2. Coordination des activités de l'équipe et suivi de l'avancement.
 3. Évaluation des performances du modèle et ajustements nécessaires.
 4. Collaboration avec l'équipe pour optimiser les algorithmes et améliorer la précision du modèle.

En tant qu'équipe, nous avons collaboré pour concevoir et mettre en place une architecture robuste pour notre solution de conversion du langage naturel en requêtes SQL. Notre principal défi a été de garantir la précision, la rapidité et l'efficacité du modèle. Nous avons réalisé que pour répondre efficacement aux besoins de notre solution, nous devons concevoir une architecture capable de

traiter un grand volume de données d'entraînement tout en assurant une performance optimale du modèle.

1.4. Tâches attribuées

Depuis notre intégration à **PEAQOCK**, nous nous sommes engagés dans une auto-formation couvrant divers sujets tels que la préparation des données, les techniques de fine-tuning des modèles de langage, et les outils nécessaires pour notre projet, tels que Docker et les bases de données relationnelles. Après cette phase d'auto-formation, nous avons entrepris la collecte des données nécessaires pour l'entraînement du modèle.

Une fois les données collectées, nous avons réalisé plusieurs tentatives de fine-tuning et de tests en utilisant différentes méthodes et différents modèles. Ces étapes nous ont permis d'évaluer la performance de chaque approche et de sélectionner la meilleure configuration pour notre solution.

1.5. Contexte métier

Dans notre contexte métier, axé sur l'amélioration de l'interopérabilité des données et l'optimisation de leur utilisation, le projet OneGate de PEAQOCK utilise actuellement l'API OpenAI pour convertir les requêtes en langage naturel en requêtes SQL. Cependant, cette solution présente des préoccupations de sécurité des données, car les informations envoyées à l'API OpenAI sont stockées par OpenAI. Les clients de **OneGate** préfèrent une solution plus sécurisée et privée. Par conséquent, notre objectif est de remplacer l'API OpenAI par un modèle open-source et privé, capable de garantir la confidentialité et la sécurité des données tout en fournissant des performances de haut niveau.

1.6. Description de contenu

La structure de ce rapport est la suivante :

- Le premier chapitre présente le contexte général du projet. Ceci passe par une présentation de l'organisme d'accueil (**Peaock financials**), une brève description du travail réalisé.
- Le deuxième chapitre aborde le contexte métier de notre projet, à savoir le l'état de l'art dans le domaine des LLM, spécifiquement text-to-code llms et text-to-sql.
- Le troisième chapitre est dédié à la description des différentes technologies adoptées durant la réalisation du travail.
- Le quatrième chapitre détaille l'approche adoptée pour la réalisation de ce travail, accompagnée d'une concrétisation de celui-ci.
- Le cinquième et le dernier chapitre va présenter les défis et les futurs prospects de ce projet et qu'est-ce qu'on peut améliorer

Conclusion

En résumé, ce chapitre a fourni une vue d'ensemble de PEAQOCK, en mettant en évidence ses clients, ses valeurs fondamentales et ses solutions innovantes. Nous avons également présenté l'organisation de l'équipe de travail, les tâches attribuées, offrant ainsi un contexte clair pour comprendre notre projet.

Chapitre 2. L'état de l'art des LLM

Introduction

Dans un monde de plus en plus dominé par les données, la capacité à interagir efficacement avec les systèmes de gestion de bases de données devient cruciale pour les entreprises. Cependant, la complexité des langages de programmation et des requêtes SQL constitue souvent une barrière pour les utilisateurs non techniques. C'est dans ce contexte que les modèles de langage de grande taille (LLMs) prennent toute leur importance.

Ces modèles permettent de démocratiser l'accès aux données en offrant des interfaces en langage naturel, facilitant ainsi la prise de décisions basée sur les données sans nécessiter de compétences techniques avancées.

Les LLMs, ou modèles de langage de grande taille, représentent une avancée significative dans le domaine de l'intelligence artificielle. Ils sont capables de comprendre et de générer du texte avec une précision et une fluidité impressionnante.

Basés sur l'architecture Transformer, ces modèles utilisent des techniques de deep learning pour analyser et traiter de vastes quantités de données textuelles.

Leur capacité à apprendre les relations sémantiques et contextuelles entre les mots leur permet d'accomplir une variété de tâches, de la génération de texte à la traduction en passant par la création de code.

Dans le cadre du projet OneGate de PEAQOCK, l'utilisation de LLMs pour convertir des requêtes en langage naturel en requêtes SQL présente un intérêt particulier.

Actuellement, **OneGate** s'appuie sur l'API OpenAI pour cette tâche. Cependant, les préoccupations concernant la sécurité des données et la confidentialité poussent l'entreprise à explorer des solutions open-source et privées. Cette démarche vise à garantir que les données sensibles des clients restent protégées tout en continuant à bénéficier des avancées technologiques en matière de traitement du langage naturel.

Ce chapitre explore en profondeur le contexte technique de notre projet.

Nous commencerons par définir ce que sont les LLMs et comment ils fonctionnent. Ensuite, nous aborderons les concepts de Text-to-Code et Text-to-SQL, deux applications clés des LLMs dans notre domaine.

Enfin, nous examinerons l'état de l'art dans ces domaines, en nous appuyant sur les dernières recherches et les modèles les plus performants actuellement disponibles.

Des exemples concrets et des espaces pour des illustrations permettront de mieux comprendre ces technologies et leur application dans le projet OneGate.

2.1. Les large language models : LLMs

2.1.1. Définition des LLMs

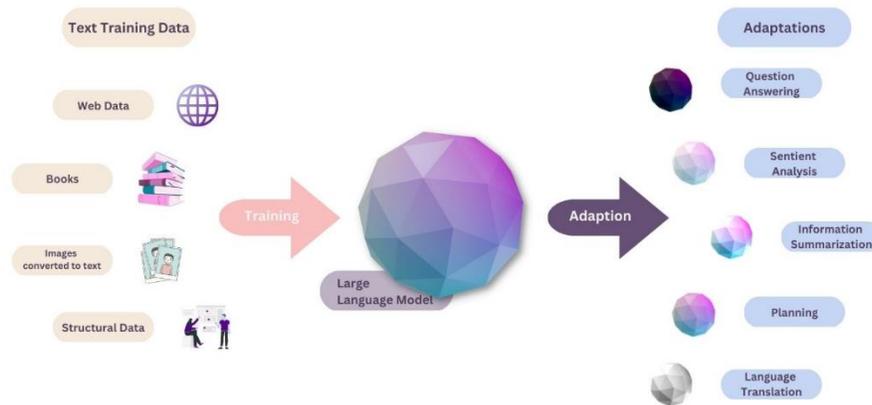


Figure 2.1: schéma qui démontre comment fonctionne les LLMs

Un large language model est un type avancé de modèle de langage qui est entraîné en utilisant des techniques d'apprentissage profond sur des quantités massives de données textuelles. Ces modèles sont capables de générer du texte semblable à celui produit par les humains et d'effectuer diverses tâches de traitement de langage naturelle.

En revanche, la définition d'un modèle de langage fait référence au concept d'attribuer des probabilités à des séquences de mots, basées sur l'analyse de corpus de textes.

Un modèle de langage peut avoir une complexité variable, allant de simples modèles n-gram à des modèles de neural networks plus sophistiqués.

Cependant, le terme "large language model" fait généralement référence à des modèles qui utilisent des techniques d'apprentissage profond et ont un grand nombre de paramètres, pouvant aller de millions à des milliards. Ces modèles d'IA peuvent capturer des modèles complexes dans le langage et produire du texte qui est souvent indiscernable de celui écrit par des humains.

2.1.2. Types des LLMs

2.1.2.1. Modèles Autorégressifs (Autoregressive Models)

Les modèles autorégressifs génèrent du texte de manière séquentielle, en prédisant chaque token en fonction des précédents. Ils excellent dans les tâches nécessitant une génération de texte cohérente et peuvent être utilisés pour diverses applications de génération de langage naturel. Ces

modèles sont polyvalents mais peuvent parfois produire des résultats répétitifs ou incohérents, en particulier pour des séquences plus longues.

Cas d'utilisation : Écriture créative, chatbots, complétion de texte, génération de code

Exemples :

- GPT-3, GPT-4
- LaMDA
- PaLM

2.1.2.2. *Modèles Encodeur-Décodeur*

Les modèles encodeur-décodeur se composent de deux éléments principaux : un encodeur qui traite l'entrée et un décodeur qui génère la sortie. Cette architecture les rend particulièrement efficaces pour les tâches impliquant la transformation d'une séquence en une autre. Ils utilisent souvent des mécanismes d'attention pour se concentrer sur les parties pertinentes de l'entrée lors de la génération.

Cas d'utilisation : Traduction automatique, résumé de texte, systèmes de questions-réponses

Exemples :

- T5 (Text-to-Text Transfer Transformer)
- BART (Bidirectional and Auto-Regressive Transformers)
- MarianMT

2.1.2.3. *Modèles de Langage Masqués (Masked Language Models)*

Les Modèles de Langage Masqués sont entraînés en prédisant des mots masqués dans une phrase, ce qui leur permet de capturer le contexte bidirectionnel. Cette approche d'entraînement les rend particulièrement efficaces pour les tâches de compréhension. Bien qu'ils excellent en compréhension, ils ne sont pas directement adaptés aux tâches de génération de texte sans ajustement fin ou modifications architecturales.

Cas d'utilisation : Analyse de sentiment, reconnaissance d'entités nommées, classification de texte

Exemples :

- BERT (Bidirectional Encoder Representations from Transformers)
- RoBERTa (Robustly Optimized BERT Approach)
- ALBERT (A Lite BERT)

2.1.2.4. *Modèles de Langage à Préfixe(Prefix Language Models)*

Ces Modèles de Langage à Préfixe combinent les caractéristiques des modèles autorégressifs et des modèles de langage masqués. Ils sont entraînés sur divers motifs d'entrée-sortie, les rendant flexibles pour gérer différents types de tâches langagières. Cette polyvalence leur permet de bien performer dans les tâches de compréhension et de génération du langage naturel.

Cas d'utilisation : Traitement du langage multi-tâches, génération de texte, analyse de sentiment

Exemples :

- UniLM (Unified Language Model)

- MASS (Masked Sequence to Sequence Pre-training)
- ERNIE (Enhanced Representation through kNowledge IntEgration)

2.1.2.5. *Multimodal Models*

Les modèles multimodaux peuvent traiter et générer du contenu à travers différentes modalités, généralement en combinant texte et images. Ils intègrent des informations visuelles et textuelles, permettant de nouveaux types d'applications d'IA nécessitant la compréhension et la génération de contenu à travers plusieurs modes de communication. Ces modèles utilisent souvent des encodeurs séparés pour différentes modalités avec un décodeur partagé ou un mécanisme de fusion.

Cas d'utilisation : Légende d'images, réponse à des questions visuelles, génération d'images à partir de texte

Exemples :

- DALL-E
- CLIP (Contrastive Language-Image Pre-training)
- VisualBERT

2.1.3. Les modèles Text-to-code et Text-to-Sql

2.1.3.1. *Text-to-code LLMs* :

Les modèles de langage de grande taille (LLMs) pour la génération de code, aussi appelés modèles text-to-code, sont des outils avancés d'intelligence artificielle capables de convertir des descriptions en langage naturel en code source. Ces modèles sont entraînés sur de vastes corpus de code provenant de divers langages de programmation et utilisent l'architecture Transformer pour comprendre et générer du code.

Les modèles text-to-code facilitent plusieurs tâches essentielles dans le développement logiciel. Ils peuvent générer du code à partir d'instructions textuelles, comme illustré dans l'exemple ci-dessous, où une instruction en langage naturel est convertie en un programme en C qui calcule la somme des nombres pairs et impairs de 1 à 50 en utilisant des boucles do-while.

❖ *Capacités des Modèles Text-to-Code* :

1. **Génération de Code** : Les LLMs peuvent générer du code complet à partir d'une simple description. Par exemple, une instruction comme "Écrire une fonction en Python pour trier une liste d'entiers" peut être automatiquement transformée en code Python fonctionnel.
2. **Complétion de Code** : Ces modèles peuvent aider à compléter du code partiellement écrit. Lorsqu'un développeur commence à écrire une fonction ou une méthode, le modèle peut suggérer les lignes suivantes, accélérant ainsi le processus de développement.
3. **Débugage** : Les LLMs peuvent également être utilisés pour détecter et corriger des erreurs dans le code. En analysant le code source, ils peuvent identifier les bugs potentiels et proposer des solutions, réduisant ainsi le temps nécessaire pour le débogage.

4. **Traduction de Code :** Ces modèles sont capables de traduire du code d'un langage de programmation à un autre. Par exemple, un programme écrit en Java peut être converti en Python, facilitant ainsi la migration de code entre différentes plateformes.
5. **Documentation Automatique :** Les modèles text-to-code peuvent générer des commentaires et de la documentation pour le code, aidant ainsi les développeurs à maintenir un code bien documenté et facile à comprendre.

Exemple de : text to code

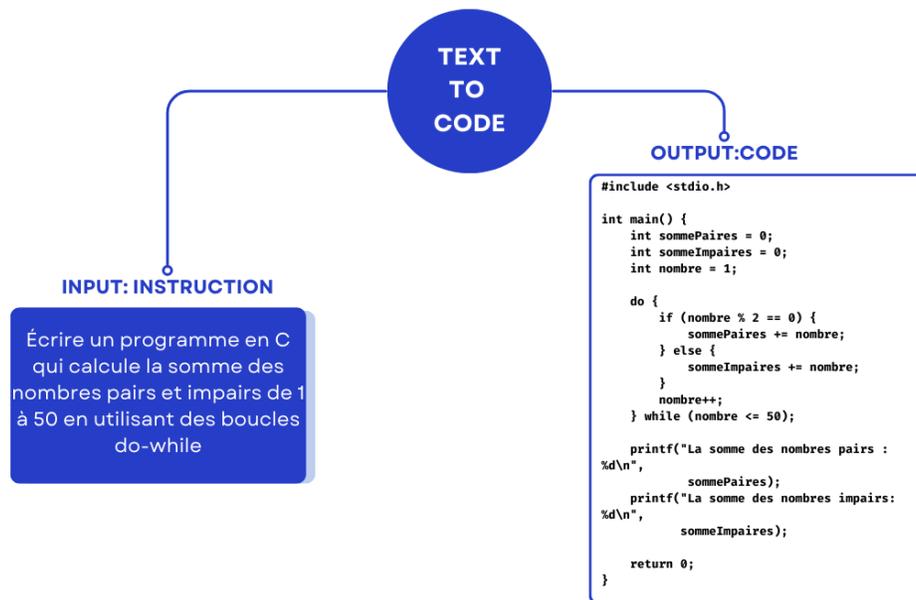


Figure 2.2: Text-to-code LLMs

Les modèles text-to-code les plus connus incluent OpenAI Codex, qui alimente GitHub Copilot, un outil qui aide les développeurs à écrire du code plus efficacement en générant des suggestions en temps réel. De plus, des modèles comme GPT-4 d'OpenAI, Bard de Google et LLaMA de Meta ont tous des versions spécialisées pour les tâches de génération de code, démontrant l'importance croissante de cette technologie dans le développement logiciel

2.1.3.1. Text-to-Sql LLMs

Les modèles pour la génération de SQL à partir de texte, également appelés modèles text-to-SQL, sont une sous-catégorie spécialisée des modèles text-to-code.

Ces modèles sont conçus pour convertir automatiquement des questions en langage naturel en requêtes SQL structurées. Cependant, contrairement aux modèles text-to-code classiques, les modèles text-to-SQL nécessitent plus d'informations que la simple instruction pour générer du code SQL. Ils ont besoin de détails supplémentaires sur la base de données, tels que le schéma de la base de données, et parfois même des données spécifiques financiers des entreprises. Ces mécanismes sont notamment liés à des mesures de gestion de résultat et de performance non financière de l'entreprise.

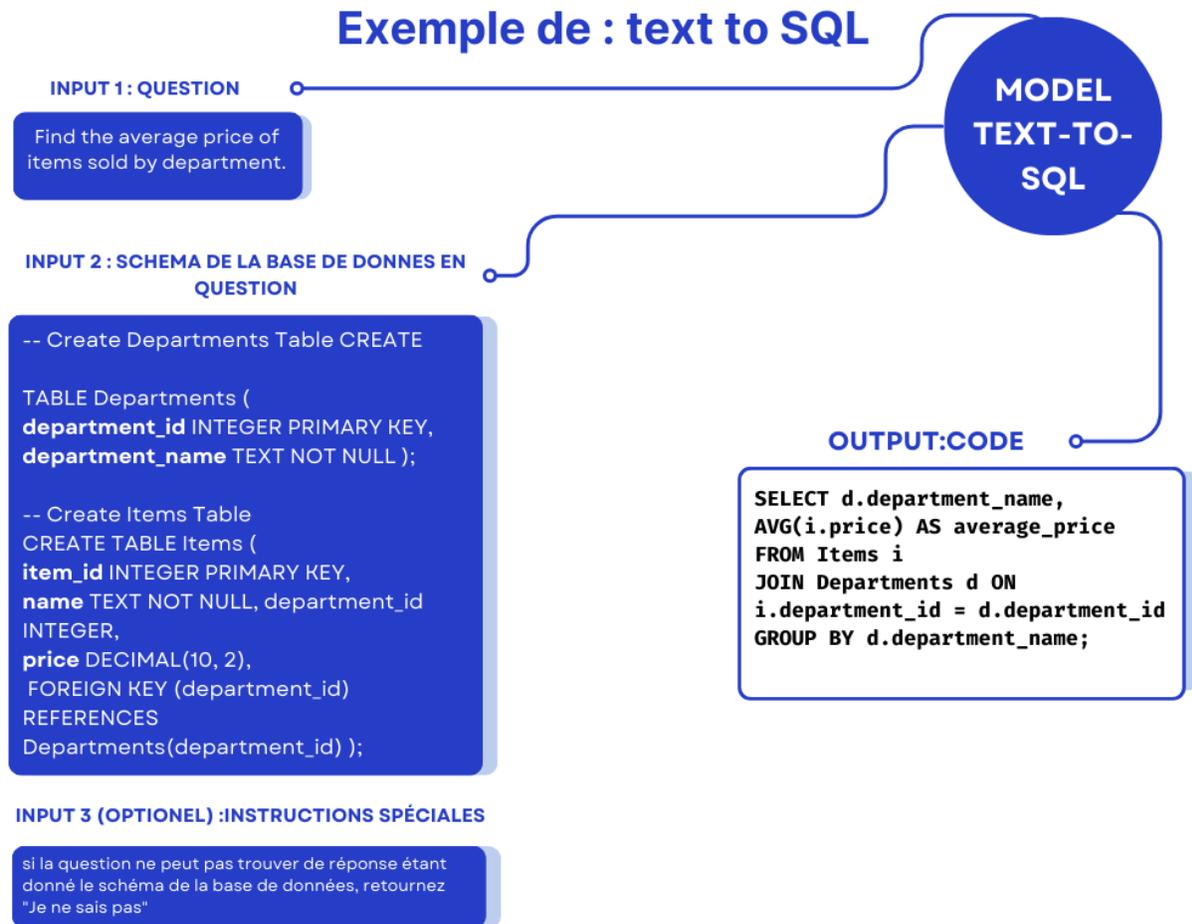


Figure 2.3: Text-to-Sql LLMs

Fonctionnement des Modèles Text-to-SQL :

Les modèles Text-to-SQL fonctionnent par un processus d'encodage et de décodage. Dans l'étape d'encodage, le modèle transforme la question en langage naturel en une représentation numérique. Ensuite, durant l'étape de décodage, cette représentation est utilisée pour générer une requête SQL qui répond à la question posée. Pour ce faire, le modèle doit comprendre le schéma de la base de données, incluant les tables, les colonnes, et les relations entre elles, afin de produire une requête SQL valide et efficace. Parfois, le modèle doit également accéder à des échantillons de données pour affiner sa compréhension et améliorer la précision de la requête générée

Exemples de Modèles Text-to-SQL

Des modèles comme GPT-4, GPT-4o, Llama3 70B instruct et SQLCoder-34B sont parmi les plus performants dans ce domaine.

On va parler de ces model et faire une comparaison dans l'axe suivant.

2.1.4. L'état de l'art

2.1.4.1. L'état de l'art dans les LLM en générale :

1. Une brève histoire des LLM :

La fondation des modèles de langage de grande taille remonte aux expériences avec les réseaux neuronaux et les systèmes de traitement de l'information neuronale menées dans les années 1950 pour permettre aux ordinateurs de traiter le langage naturel. Les chercheurs d'IBM et de l'Université de Georgetown ont collaboré pour créer un système capable de traduire automatiquement des phrases du russe vers l'anglais. Cette démonstration notable de la traduction automatique a marqué le début de la recherche dans ce domaine.

L'idée des LLMs a été évoquée pour la première fois avec la création d'Eliza dans les années 1960 : il s'agissait du premier chatbot au monde, conçu par le chercheur du MIT Joseph Weizenbaum. Eliza a marqué le début de la recherche en traitement du langage naturel (NLP), posant les bases des futurs LLMs plus complexes.

Près de 30 ans plus tard, en 1997, les réseaux Long Short-Term Memory (LSTM) ont vu le jour. Leur avènement a permis de développer des réseaux neuronaux plus profonds et plus complexes capables de traiter des quantités de données plus importantes. La suite CoreNLP de Stanford, introduite en 2010, a été l'étape suivante de la croissance, permettant aux développeurs d'effectuer des analyses de sentiments et la reconnaissance d'entités nommées.

Par la suite, en 2011, une version plus petite de Google Brain est apparue avec des fonctionnalités avancées telles que les embeddings de mots, qui ont permis aux systèmes NLP de mieux comprendre le contexte. Ce fut un tournant significatif, avec l'apparition des modèles transformateurs en 2017. Pensez à GPT, qui signifie Generative Pre-trained Transformer, capable de générer ou de "décoder" du nouveau texte. Un autre exemple est BERT - Bidirectional Encoder Representations from Transformers. BERT peut prédire ou classer le texte d'entrée en fonction des composants de l'encodeur.

À partir de 2018, les chercheurs se sont concentrés sur la construction de modèles de plus en plus grands. En 2019, les chercheurs de Google ont introduit BERT, le modèle bidirectionnel à 340 millions de paramètres (le troisième plus grand modèle de son genre) capable de déterminer le contexte et de s'adapter à diverses tâches. En pré-entraînant BERT sur une grande variété de données non structurées via l'apprentissage auto-supervisé, le modèle a pu comprendre les relations entre les mots. En peu de temps, BERT est devenu l'outil incontournable pour les tâches de traitement du langage naturel. En fait, BERT était à l'origine de chaque requête en anglais administrée via Google Search.

2. L'essor des transformateurs et de ChatGPT :

Alors que BERT continuait de se perfectionner, le GPT-2 d'OpenAI, avec 1,5 milliard de paramètres, a réussi à produire des textes convaincants. Ensuite, en 2020, ils ont lancé GPT-3 avec 175 milliards de paramètres, établissant la norme pour les LLMs et formant la base de ChatGPT. C'est avec la sortie de ChatGPT en novembre 2022 que le grand public a vraiment

commencé à prendre conscience de l'impact des LLMs. Même les utilisateurs non techniques pouvaient interagir avec le LLM, recevoir une réponse rapide et poursuivre une conversation, suscitant à la fois enthousiasme et appréhension. Principalement, c'est le modèle transformateur avec une architecture d'encodeur-décodeur qui a stimulé la création de LLMs plus grands et plus complexes, servant de catalyseur pour GPT-3 d'OpenAI, ChatGPT, et d'autres. En tirant parti de deux composants clés : les embeddings de mots (qui permettent à un modèle de comprendre les mots dans leur contexte) et les mécanismes d'attention (qui permettent à un modèle d'évaluer l'importance des mots ou des phrases), les transformateurs ont été particulièrement utiles pour déterminer le contexte. Ils ont révolutionné le domaine en raison de leur capacité à traiter de grandes quantités de données en une seule fois. Plus récemment, OpenAI a introduit GPT-4, estimé à un billion de paramètres - cinq fois la taille de GPT-3 et environ 3 000 fois la taille de BERT lors de sa première sortie. Nous couvrons l'évolution de la série GPT plus en détail plus tard.

Voici une chronologie simple du développement des LLMs, de Eliza à GPT d'OpenAI :

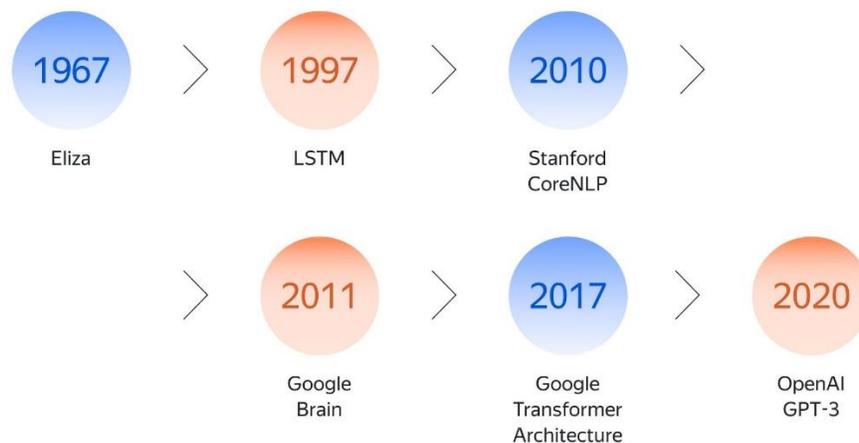


Figure 2.4 Histoire des LLMs

3. Applications des LLMs :

Les LLMs ont une multitude d'applications qui incluent la synthèse de différents textes, la création d'outils de recherche numérique plus efficaces et la fonction de chatbots. Voici quatre applications clés que nous allons examiner de plus près :

- a. **Évolution de l'IA conversationnelle** : Les LLMs ont prouvé leur capacité à générer des réponses pertinentes et cohérentes tout au long d'une conversation. Pensez aux chatbots et aux assistants virtuels. De même, les LLMs aident à rendre les systèmes de reconnaissance vocale plus précis. Soyez attentifs aux nouvelles applications qui ne manqueront pas d'émerger dans un avenir proche grâce à cela !
- b. **Compréhension des sentiments dans les textes** : Les LLMs excellent dans l'analyse des sentiments et l'extraction d'informations subjectives telles que les émotions et les

opinions. Les applications incluent les retours clients et l'analyse des réseaux sociaux ainsi que la surveillance de la marque.

- c. **Traduction automatique efficace** : Grâce aux LLMs, les systèmes de traduction deviennent plus efficaces et précis. En supprimant les barrières linguistiques, les LLMs rendent possible le partage de connaissances et la communication entre les humains partout dans le monde.
- d. **Création de contenu textuel** : Ce qui est encore plus impressionnant, c'est que les LLMs peuvent générer diverses formes de texte, comme des articles de presse et des descriptions de produits. Ils se sont même essayés à l'écriture créative avec un succès notable - à la grande joie de nombreux étudiants (et à la grande horreur de nombreux enseignants) lors de la soumission de leurs travaux de fin d'année !

4. *Limitations et défis actuels* :

Malgré les incroyables avancées dans les domaines de l'intelligence artificielle, des modèles d'apprentissage automatique et des LLMs en général, de nombreux défis restent à relever. La désinformation, les logiciels malveillants, les contenus discriminatoires, le plagiat et les informations simplement fausses peuvent conduire à des résultats non intentionnels ou dangereux ; cela remet en question ces modèles.

De plus, lorsque des biais sont introduits par inadvertance dans des produits basés sur les LLMs comme GPT-4, ils peuvent sembler "certains mais erronés" sur certains sujets. C'est un peu comme lorsqu'un politicien parle de quelque chose qu'il ne connaît pas. Surmonter ces limitations est essentiel pour instaurer la confiance du public envers cette nouvelle technologie. C'est là que le RLHF (Apprentissage par Renforcement avec Feedback Humain) entre en jeu pour aider à contrôler ou orienter les systèmes d'IA à grande échelle.

Voici quelques-uns des principaux problèmes à aborder :

- a. **Préoccupations éthiques et de confidentialité** Actuellement, il n'existe pas beaucoup de lois ou de garde-fous réglementant l'utilisation des LLMs, et comme les grands ensembles de données contiennent beaucoup de données confidentielles ou sensibles (y compris le vol de données personnelles, les violations de droits d'auteur et de propriété intellectuelle, etc.), cela soulève des questions éthiques, de confidentialité et même psychologiques parmi les utilisateurs recherchant des réponses sur des forums générés par l'IA.
- b. **Biais et préjugés** Étant donné que les LLMs sont formés sur différentes sources, ils peuvent reproduire inconsciemment les biais présents dans ces sources. Les biais culturels, raciaux, de genre, etc., peuvent être reproduits. Ces préjugés ont des conséquences réelles, telles que les décisions d'embauche, les soins médicaux ou les résultats financiers.
- c. **Coûts environnementaux et computationnels** Former un modèle de langage de grande taille nécessite une énorme puissance de calcul, ce qui impacte la consommation

d'énergie et les émissions de carbone. En outre, cela coûte cher ! De nombreuses entreprises, en particulier les plus petites, ne peuvent tout simplement pas se le permettre.

Pour contrer ces effets potentiellement nocifs, les chercheurs se concentrent sur la conception des LLMs autour de trois piliers principaux : l'utilité, la véracité et l'innocuité. Si un LLM peut respecter ces trois principes, il est considéré comme "aligné" - un terme qui comporte des éléments de subjectivité. Le RLHF offre une solution utile dans ce cas.

En résumé, bien que les LLMs présentent un potentiel énorme pour révolutionner divers domaines, il est crucial de surmonter les défis actuels pour garantir leur utilisation éthique, équitable et durable

5. *Quelques benchmarks et comparaisons :*

Pour fournir une analyse approfondie, Nous allons examiner les performances des différents modèles à travers des benchmarks comme Spider, BIRD et d'autres, en comparant les LLMs open-source et propriétaires.

Cette analyse nous permettra de mieux comprendre les forces et les faiblesses de chaque type de modèle et d'informer les décisions futures en matière de développement et de déploiement de LLMs dans des environnements sécurisés et efficaces.

- **Classement des LLMs Open (Juillet 2024) :**

Pour évaluer les performances des modèles de langage de grande taille (LLMs) open-source, nous nous référons au classement de Hugging-Face, disponible sur [open-llm-leaderboard](https://open-llm-leaderboard.com). Ce classement compare divers modèles de LLMs en utilisant plusieurs benchmarks pour fournir une évaluation précise de leurs capacités.

La liste des métriques :

- **Average (Moyenne) :** La moyenne des performances du modèle sur tous les benchmarks, indiquant une vue d'ensemble de ses capacités globales.
- **IFEval (Évaluation IFEval) :** Une mesure de la performance du modèle sur un ensemble de tâches d'évaluation standardisées utilisées pour évaluer l'intelligence artificielle.
- **BBH (Haut Niveau de Compréhension de Base) :** Une évaluation de la capacité du modèle à comprendre et à répondre correctement aux questions de compréhension de base.
- **MATH Lvl 5 (Niveau Math 5) :** Une mesure des performances du modèle dans la résolution de problèmes mathématiques de niveau 5, indiquant sa capacité à traiter des questions de mathématiques de complexité modérée.
- **GPQA (Évaluation de Qualité des Questions Générales) :** Une évaluation de la qualité des réponses du modèle aux questions générales, testant sa capacité à fournir des informations précises et pertinentes.

- **MUSR (Évaluation des Réponses de Compréhension)** : Une mesure de la capacité du modèle à fournir des réponses de compréhension détaillées et précises.
- **MMLU-PRO (Compréhension de Niveau Professionnel)** : Une évaluation de la performance du modèle dans des tâches nécessitant une compréhension professionnelle et des connaissances spécialisées.

Model	Average	IFEval	BBH	MATH Lv1 5	GPQA	MUSR	MMLU-PRO
Owen/Owen2-72B-Instruct	42.49	79.89	57.48	35.12	16.33	17.17	48.92
meta-llama/Meta-llama-3-70B-Instruct	36.18	80.99	50.19	23.34	4.92	10.92	46.74
Owen/Owen2-72B	35.13	38.24	51.86	29.15	19.24	19.73	52.56
mistralai/Mixtral-8x22B-Instruct-v0.1	33.89	71.84	44.11	18.73	16.44	13.49	38.7
HuggingFaceH4/zephyr-orpo-141b-A35b-v0.1	33.77	65.11	47.5	18.35	17.11	14.72	39.85
microsoft/Phi-3-medium-4k-instruct	32.67	64.23	49.38	16.99	11.52	13.05	40.84
01-ai/Yi-1.5-34B-Chat	32.63	60.67	44.26	23.34	15.32	13.06	39.12
CohereForAI/c4ai-command-r-plus	30.86	76.64	39.92	7.55	7.38	20.42	33.24
internlm/internlm2_5-7b-chat	30.46	61.4	57.67	8.31	10.63	14.35	30.42
Owen/Owen1.5-110B	29.56	34.22	44.28	23.04	13.65	13.71	48.45
abacusai/Smaug-72B-v0.1	29.56	51.7	42.42	17.75	9.62	15.39	40.46

Figure 2.5 : Classement des LLM Open

- **Classement des LLMs Open et Closed :**

Et pour évaluer les LLM y compris les modèles privés nous référons à un autre classement : **LMSYS Chatbot Arena Leaderboard** disponible sur huggingface, ce classement se base sur les éléments suivants :

- **Score de l'arène (Arena Score)** : Il s'agit de la note Elo du modèle, qui mesure sa performance basée sur des comparaisons par paires. Des scores plus élevés indiquent une meilleure performance.
- **IC à 95% (95% CI, Intervalle de Confiance)** : Cela indique la plage dans laquelle le véritable score de l'arène est susceptible de se situer, avec une confiance de 95%. Il est présenté sous forme de plage (par exemple, +3/-3), montrant la variation possible au-dessus et en dessous du score rapporté.
- **Votes (Votes)** : Le nombre de comparaisons par paires humaines (votes) qui ont été exprimées pour le modèle. Plus il y a de votes, plus le score est généralement fiable.
- **Organisation (Organization)** : L'entité ou l'entreprise qui a développé ou publié le modèle.
- **Licence (License)** : Le type de licence sous laquelle le modèle est publié, ce qui peut indiquer ses restrictions d'utilisation (par exemple, propriétaire, open source).

Rank* (UB)	Model	Arena Score	95% CI	Votes	Organization	License	Knowledge Cutoff
1	GPT-4o-2024-05-13	1287	+3/-3	56905	OpenAI	Proprietary	2023/10
2	Claude 3.5 Sonnet	1272	+4/-4	24913	Anthropic	Proprietary	2024/4
2	Gemini-Advanced-0514	1267	+3/-3	42981	Google	Proprietary	Online
3	Gemini-1.5-Pro-API-0514	1262	+3/-3	49828	Google	Proprietary	2023/11
4	Gemini-1.5-Pro-API-0409-Preview	1258	+3/-3	55567	Google	Proprietary	2023/11
4	GPT-4-Turbo-2024-04-09	1257	+3/-4	72512	OpenAI	Proprietary	2023/12
6	GPT-4-1106-preview	1251	+3/-3	86474	OpenAI	Proprietary	2023/4
7	Claude 3 Opus	1248	+2/-2	143189	Anthropic	Proprietary	2023/8
8	GPT-4-0125-preview	1246	+3/-2	79732	OpenAI	Proprietary	2023/12
9	Yi-Large-preview	1241	+3/-4	47636	01 AI	Proprietary	Unknown

Figure 2.6: LMSYS Chatbot Arena Leaderboard

2.1.4.2. L'état de l'art dans les LLM Text-to-code :

Dans cette partie, nous explorerons certains des meilleurs LLM de génération de code de 2024, en examinant leurs caractéristiques, leurs points forts et comment ils se comparent les uns aux autres.

- **OpenAI Codex :**

OpenAI Codex, un descendant de GPT-3, est un modèle d'IA puissant qui génère du code à partir du langage naturel. Il alimente GitHub Copilot, offrant une autocomplétion dans divers IDE. Codex peut créer du code dans plusieurs langages, avec une maîtrise particulière du Python. Il a été entraîné sur un vaste corpus de code provenant de millions de dépôts GitHub, lui permettant de générer des extraits de code pertinents. En plus de la génération de code, Codex aide à la transpilation, à l'explication du code et à la refactorisation.

OpenAI Codex vs GitHub Copilot : GitHub Copilot est une application directe de Codex, bien intégrée dans les flux de travail des développeurs. Codex, cependant, offre une gamme plus large d'applications et peut être intégré dans des outils personnalisés.

OpenAI Codex vs ChatGPT : Codex se spécialise dans la génération de code, tandis que ChatGPT excelle dans la génération de texte conversationnel. La formation spécialisée de Codex en fait un meilleur choix pour la génération de code précise.

- **Code Llama :**

Code Llama est l'outil de programmation IA de Meta, conçu pour aider les développeurs à générer du code. Avec 70 milliards de paramètres, il comprend des tâches de programmation complexes et génère des extraits de code. Son entraînement sur un ensemble de données diversifié lui permet de travailler avec divers langages de programmation et frameworks.

Code Llama vs GPT-4: Code Llama est spécifiquement optimisé pour la génération de code, ce qui le rend potentiellement plus efficace pour cette tâche par rapport à GPT-4, qui a des capacités plus générales.

Code Llama vs Copilot Code : Llama, avec son modèle de 70 milliards de paramètres, suggère une capacité de génération de code plus puissante que Copilot, qui offre des suggestions en temps réel au sein d'un IDE.

- **Codet5 et Codet5+ :**

Codet5 et Codet5+ sont des outils de programmation assistés par IA développés par Salesforce. Ils font partie d'une famille de modèles de langage encodeur-décodeur conçus pour la génération et la compréhension de code. Ils sont formés sur un ensemble de données diversifié, ce qui leur permet de travailler avec divers langages de programmation et frameworks. Ils peuvent comprendre des tâches de programmation complexes, générer des extraits de code et fournir des solutions aux problèmes de codage.

Codet5 est basé sur le framework T5 de Google, mais intègre une meilleure connaissance spécifique au code. Il peut effectuer des opérations comme l'autocomplétion de code, la summarisation et la traduction entre différents langages de programmation. Malgré ses capacités, Codet5 n'est pas aussi largement disponible que d'autres outils de programmation assistés par IA comme GitHub Copilot ou OpenAI Codex.

Codet5+ est une version améliorée de Codet5, avec une architecture de modèle flexible et des objectifs d'apprentissage diversifiés. Il peut fonctionner comme un modèle uniquement encodeur, uniquement décodeur ou unifié, selon la tâche. Codet5+ a atteint des performances de pointe sur de nombreuses tâches difficiles, y compris l'évaluation zéro-shot sur le benchmark de génération de code HumanEval.

Codet5 vs Codex : Codet5 est plus abordable et atteint des performances de pointe sur de nombreuses tâches de génération de code. Codex, alimentant GitHub Copilot, fournit des suggestions de codage en temps réel au sein d'un IDE.

- **Tabnine**

Tabnine est un outil d'auto-complétion de code alimenté par l'IA, utilisant un modèle propriétaire formé sur un large éventail de dépôts de code de haute qualité. Il prend en charge tous les principaux langages de programmation et s'intègre à la plupart des IDE modernes.

Tabnine vs Copilot : Tabnine et GitHub Copilot offrent tous deux une auto-complétion de code alimenté par l'IA. Copilot fournit des suggestions en temps réel, tandis que Tabnine peut être personnalisé en fonction du code d'une organisation, offrant des suggestions plus personnalisées.

Tabnine vs ChatGPT : ChatGPT peut générer du code à partir de requêtes en langage naturel mais nécessite des instructions détaillées. Tabnine est conscient des pratiques de codage organisationnelles, ce qui se reflète dans la précision des suggestions de code.

En plus de ces modèles, d'autres existent, tels que :

- **StarCoder** : Développé par Huggingface, ce modèle est conçu pour assister les développeurs avec une large gamme de tâches de codage.
- **Polycoder** : Un outil open-source de génération de code développé par Carnegie Mellon, particulièrement performant en langage C.
- **Codeium** : Un modèle d'IA avancé offrant des fonctionnalités d'autocomplétion et de génération de code.
- **Amazon CodeWhisperer** : Fournissant des suggestions en temps réel pour améliorer la qualité du code.
- **Deepseek Coder** : Optimisé pour générer et améliorer du code en Python, Java, et C++.
- **Pangu-Coder2** : Un modèle avancé spécialisé dans les tâches de codage diverses.

Ces modèles diversifient les options disponibles pour les développeurs, chacun offrant des fonctionnalités uniques pour répondre à des besoins spécifiques en matière de génération de code.

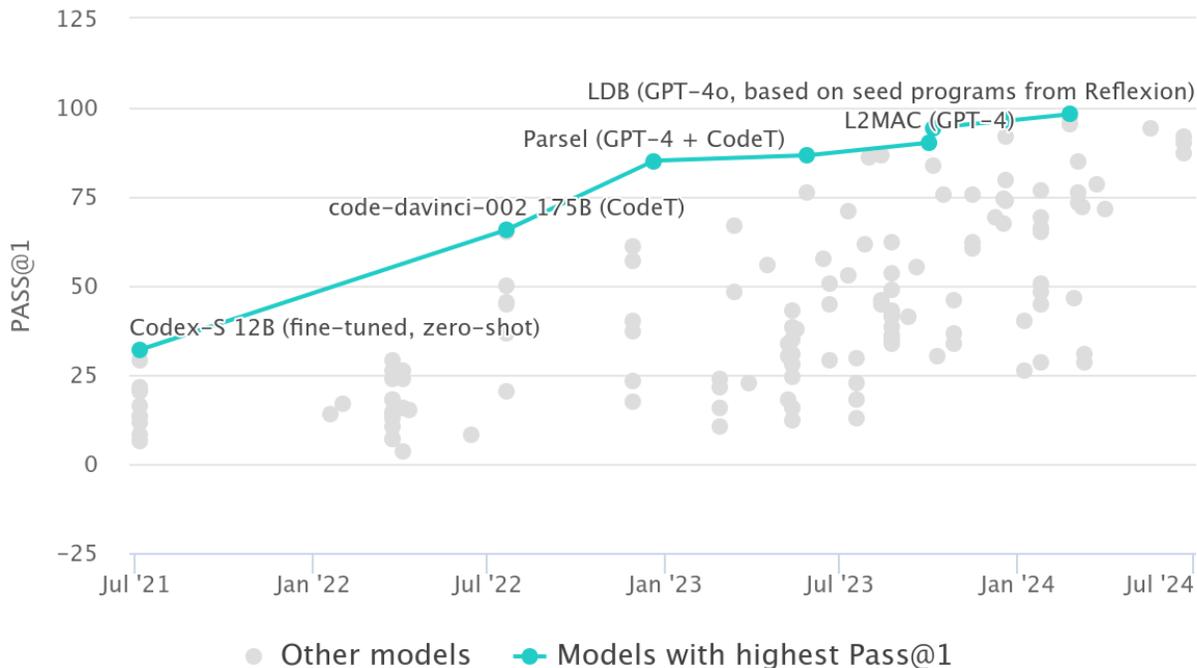


Figure 2.7: Classement des LLM de Génération de code sur le benchmark HumanEval

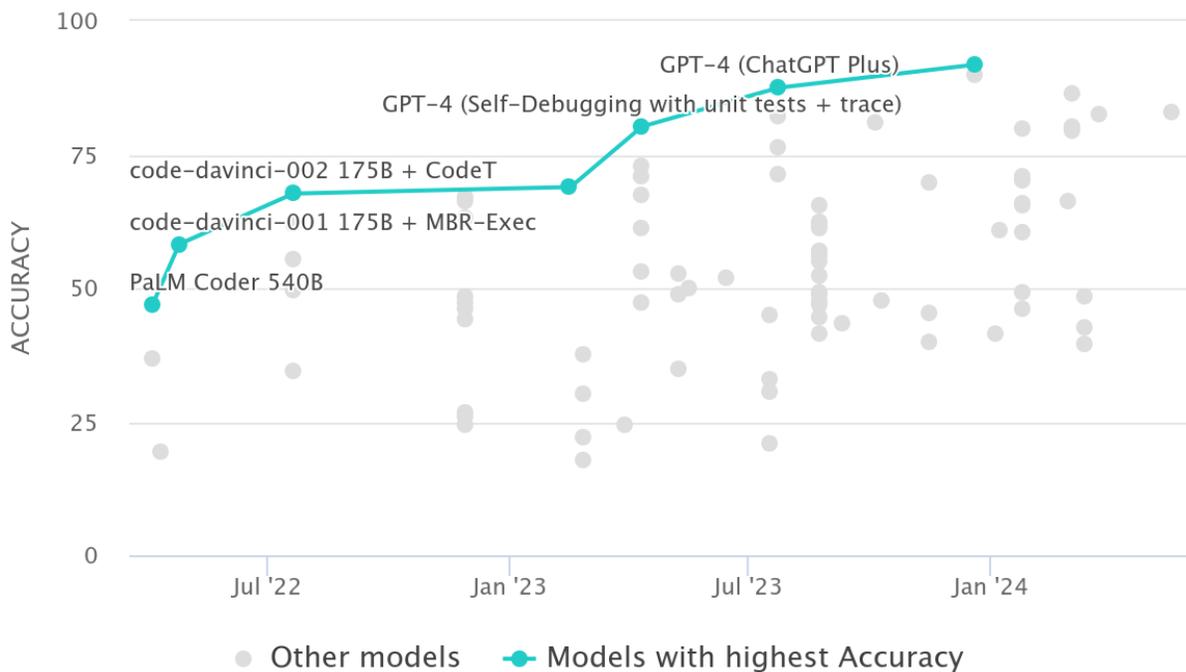


Figure 2.8: Classement des LLM de Génération de code sur le benchmark MBPP

- **Conclusion :**

Ces merveilles de l'IA ne sont pas seulement des outils, mais des portes d'entrée vers des domaines inexplorés d'efficacité et de créativité. De l'automatisation des tâches de codage routinières dans le développement logiciel à l'innovation dans des domaines comme l'analyse de données et la recherche en IA, leur impact est profond. Ces modèles permettent aux développeurs de créer des applications plus robustes et sans erreur, plus rapidement que jamais. L'avenir du codage avec les LLM semble illimité. Nous nous dirigeons vers un monde où le codage est plus accessible, plus intuitif et, surtout, plus aligné sur notre façon humaine de penser et de résoudre les problèmes. La barrière entre l'idée et la mise en œuvre s'amincit, ouvrant un univers de possibilités.

2.1.4.3. *L'état de l'art dans les LLM Text-to-Sql :*

La génération précise de SQL à partir de questions en langage naturel (text-to-SQL) est un problème de longue date, car il est difficile de comprendre les questions des utilisateurs, d'appréhender le schéma de la base de données et de générer du SQL.

Les systèmes text-to-SQL conventionnels incluent l'ingénierie humaine et les réseaux de neurones profonds.

Par la suite, des modèles de langage pré-entraînés (PLM) ont été développés et utilisés pour les tâches de text-to-SQL, obtenant des performances prometteuses.

Alors que les bases de données modernes deviennent plus complexes et les questions des utilisateurs plus difficiles, les PLM aux capacités de compréhension limitées peuvent conduire à une génération incorrecte de SQL. Cela nécessite des méthodes d'optimisation plus sophistiquées et sur mesure, ce qui, à son tour, restreint les applications des systèmes basés sur les PLM. Plus récemment, les grands modèles de langage (LLM) ont démontré des capacités significatives de compréhension du langage naturel à mesure que l'échelle du modèle continue d'augmenter. Par conséquent, l'intégration de l'implémentation basée sur les LLM peut apporter des opportunités, des défis et des solutions uniques à la recherche text-to-SQL.

La mise en œuvre du text-to-SQL basé sur les LLM peuvent être divisés en 3 aspects :

- 1. Compréhension de la question :** La question en langage naturel est une représentation sémantique de l'intention de l'utilisateur, avec laquelle la requête SQL générée correspondante est censée s'aligner.
- 2. Compréhension du schéma :** Le schéma fournit la structure des tables et des colonnes de la base de données, et le système text-to-SQL doit identifier les composants cibles dans la base de données qui correspondent à la question de l'utilisateur.
- 3. Génération de SQL :** Cela implique d'incorporer l'analyse ci-dessus puis d'écrire une syntaxe correcte pour générer des requêtes SQL exécutables qui peuvent récupérer la réponse souhaitée. Les LLM ont prouvé qu'ils effectuent une bonne implémentation de base, bénéficiant d'une analyse sémantique plus puissante rendue possible par un corpus d'entraînement plus riche par rapport aux PLM.

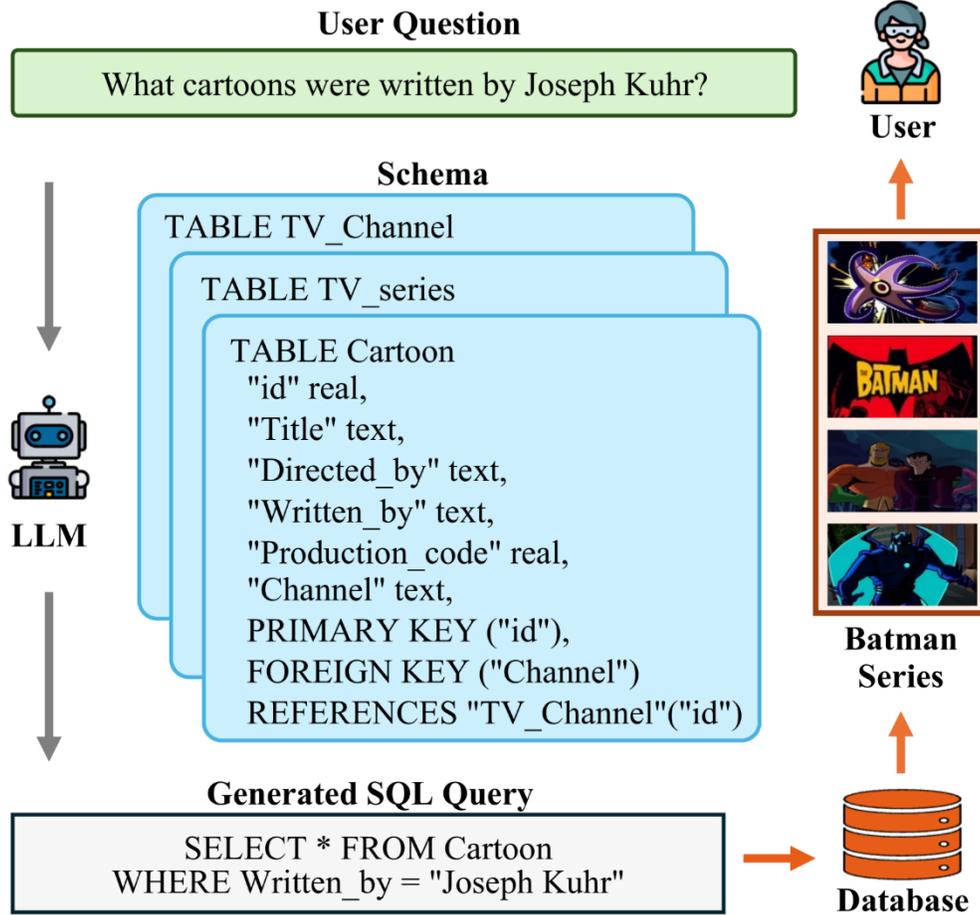


Figure 2.9: Un exemple de texte vers SQL basé sur LLM sélectionné dans l'ensemble de données Spider

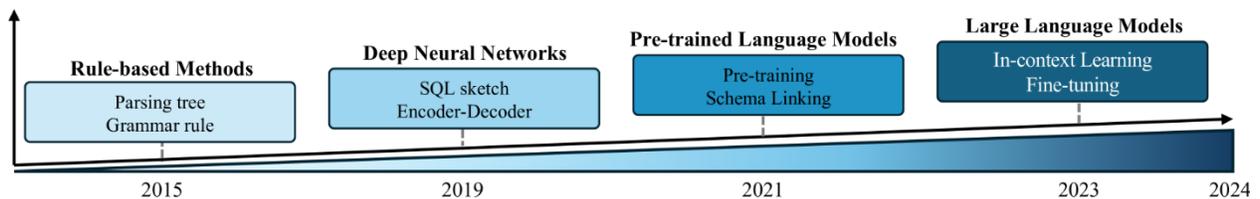


Figure 2.10: le processus évolutif de la recherche Text-to-SQL

Nous allons maintenant examiner les classements des modèles de text-to-SQL basés sur 2 jeux de données de référence **BIRD** (BIG Bench for LaRge-scale Database Grounded Text-to-SQL Evaluation) et **spider**.

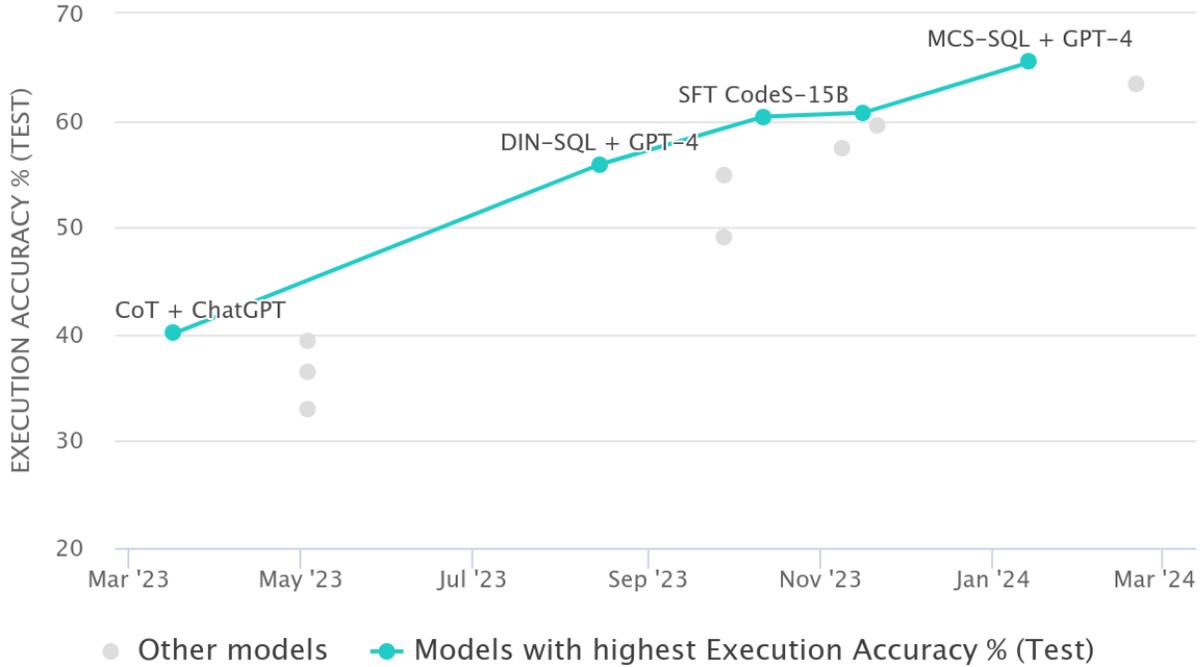


Figure 2.11: Text-To-SQL on BIRD (BIG Bench for LaRge-scale Database Grounded Text-to-SQL Evaluation)

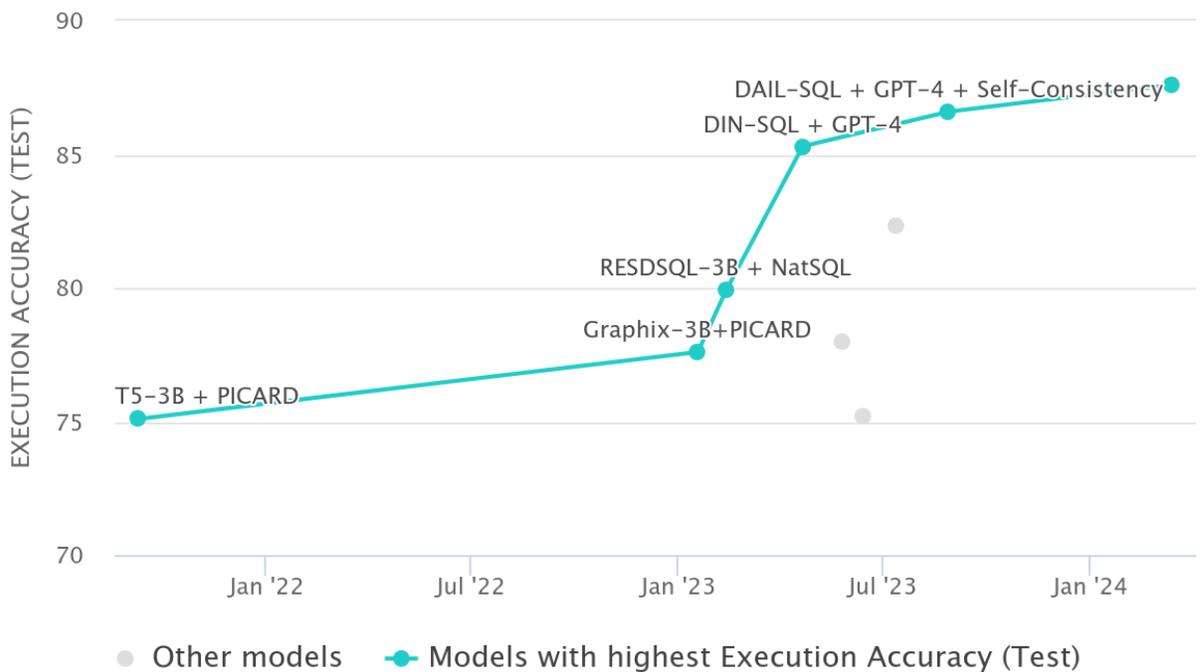


Figure 2.12: Text-To-SQL benchmark on spider

Rank	Model	Execution Accuracy (Test)	Exact Match Accuracy (Test)	Execution Accuracy (Dev)	Exact Match Accuracy (Dev)	Paper	Code	Result	Year
1	PET-SQL	87.6	66.6			PET-SQL: A Prompt-Enhanced Two-Round Refinement of Text-to-SQL with Cross-consistency			2024
2	DAIL-SQL + GPT-4 + Self-Consistency	86.6		84.4	74.4	Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation			2023
3	DIN-SQL + GPT-4	85.3	60			DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction			2023
4	C3 + ChatGPT + Zero-Shot	82.3		81.8		C3: Zero-shot Text-to-SQL with ChatGPT			2023
5	RESDSL-3B + NatSQL	79.9	72.0	84.1	80.5	RESDSL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL			2023
6	T5-3B+NatSQL+Token Preprocessing	78		73.7	69.4	Improving Generalization in Language Model-Based Text-to-SQL Semantic Parsing: Two Simple Semantic Boundary-Based Techniques			2023
7	Graphix-3B+PICARD	77.6		81.0	77.1	Graphix-T5: Mixing Pre-Trained Transformers with Graph-Aware Layers for Text-to-SQL Parsing			2023
8	T5-SR	75.2	72.4		77.2	T5-SR: A Unified Seq-to-Seq Decoding Strategy for Semantic Parsing			2023

Figure 2.13: classement des model LLM dans la tache Text to Sql

En conclusion, le domaine du text-to-SQL a connu une évolution significative, passant des systèmes conventionnels aux modèles de langage pré-entraînés (PLM), et plus récemment aux grands modèles de langage (LLM). Cette progression a permis de relever les défis croissants posés par la complexité des bases de données modernes et la sophistication des requêtes des utilisateurs.

Les LLM ont démontré des capacités impressionnantes dans la compréhension du langage naturel, offrant de nouvelles opportunités pour améliorer la précision et l'efficacité des systèmes text-to-SQL. Leur approche en trois étapes - compréhension de la question, compréhension du schéma, et génération de SQL - permet une analyse plus fine et une meilleure adaptation aux intentions de l'utilisateur.

Les benchmarks BIRD et Spider ont fourni des plateformes essentielles pour évaluer et comparer les performances des différents modèles text-to-SQL. Ces évaluations ont mis en lumière les progrès significatifs réalisés grâce à l'utilisation des LLM, tout en soulignant les domaines nécessitant encore des améliorations.

Alors que la recherche dans ce domaine continue d'avancer, nous pouvons nous attendre à des développements encore plus prometteurs, ouvrant la voie à des interactions plus naturelles et plus efficaces entre les utilisateurs et les bases de données complexes. L'avenir du text-to-SQL semble

prometteur, avec le potentiel de transformer radicalement la manière dont nous interagissons avec les données structurées.

Conclusion

En conclusion, les modèles de langage de grande taille (LLMs) ont démontré leur potentiel exceptionnel dans la génération de code et les requêtes SQL. Ils permettent de convertir des instructions en langage naturel en code source ou en requêtes SQL avec une précision remarquable, simplifiant ainsi les processus de développement logiciel et l'interaction avec les bases de données. Les LLMs tels que OpenAI Codex pour la génération de code et GPT-4 pour les requêtes SQL illustrent parfaitement ces capacités avancées.

Dans ce chapitre, nous avons exploré la définition des LLMs, leurs types, et plus spécifiquement les modèles text-to-code et text-to-SQL. Nous avons discuté de leur fonctionnement, de leurs capacités et de leur état de l'art, en soulignant comment ils transforment les pratiques de développement et de gestion des données.

Dans le prochain chapitre, nous aborderons les outils et technologies utilisés dans notre projet. Nous détaillerons les environnements de développement, les frameworks, et les bibliothèques qui ont été essentiels pour la mise en œuvre et l'optimisation de notre solution basée sur les LLMs.

Chapitre 3. Outils et Technologies utilisés

Introduction

Dans ce chapitre, nous allons passer en revue les principales technologies et outils utilisés dans notre projet de Text-to-SQL basé sur les LLMs. Pour chaque élément, nous fournirons une brève définition, mettant en lumière son rôle spécifique dans notre développement. Cette approche permettra de donner un aperçu clair et concis de notre stack technologique, tout en soulignant l'importance de chaque composant dans la réalisation de notre solution.

3.1. Language, Frameworks et bibliothèques

3.1.1. Language de programmations

3.1.1.1. Python



Figure 3.1 : logo python

Python est un langage de programmation de haut niveau, interprété et polyvalent, créé par Guido van Rossum et lancé en 1991. Il se distingue par sa syntaxe claire et lisible, qui met l'accent sur la simplicité et la facilité d'utilisation.

Utilisation en IA :

Python est devenu le langage de prédilection pour le développement en intelligence artificielle et en apprentissage automatique pour plusieurs raisons :

- **Simplicité et lisibilité** : Sa syntaxe claire permet aux chercheurs et développeurs de se concentrer sur la résolution de problèmes complexes plutôt que sur les détails de la programmation.
- **Riche écosystème de bibliothèques** : Python dispose d'une vaste collection de bibliothèques spécialisées pour l'IA et l'apprentissage automatique, telles que TensorFlow, PyTorch, scikit-learn, et NumPy, qui facilitent le développement de modèles sophistiqués.
- **Communauté active** : Une large communauté de développeurs contribue constamment à l'amélioration des outils et bibliothèques, offrant un support précieux et des ressources abondantes.
- **Flexibilité** : Python s'intègre facilement avec d'autres langages et technologies, ce qui est crucial pour les projets d'IA complexes.
- **Prototypage rapide** : La nature interprétée de Python permet un développement et un test rapides des idées, ce qui est essentiel dans la recherche en IA.

Ces caractéristiques font de Python un choix privilégié pour le développement en IA, de la recherche fondamentale aux applications industrielles.

Google Colab

3.1.1.2. SQL (Structured Query Language) :

```

SELECT column1,
        AGG(column2) AS alias_name, column1
FROM table_name1 AS t1
WHERE WHERE_condition
GROUP BY column1
HAVING HAVING_condition
ORDER BY AGG(column2) DESC
JOIN table_name2 AS t2
ON t1.column2 = t2.column3;

```

Figure 3.2: language SQL

SQL est un langage de programmation standardisé conçu pour gérer et manipuler des bases de données relationnelles. Créé dans les années 1970, SQL permet aux utilisateurs d'interagir avec des bases de données pour stocker, récupérer, mettre à jour et supprimer des données.

Position dans un projet de text-to-SQL :

Dans un projet de text-to-SQL, SQL occupe une position centrale :

- **Langage cible** : SQL est le langage vers lequel les questions en langage naturel sont traduites. Le but principal du projet est de générer des requêtes SQL correctes et efficaces à partir de ces questions.
- **Interaction avec la base de données** : Les requêtes SQL générées sont utilisées pour interroger directement la base de données et obtenir les informations demandées par l'utilisateur.
- **Évaluation et validation** : La précision du modèle text-to-SQL est souvent évaluée en comparant les requêtes SQL générées avec des requêtes de référence écrites manuellement.
- **Compréhension du schéma** : La connaissance de SQL est essentielle pour comprendre et interpréter correctement les schémas de bases de données, un aspect crucial du processus de text-to-SQL.
- **Optimisation des requêtes** : Une bonne compréhension de SQL permet d'optimiser les requêtes générées pour de meilleures performances.

En résumé, SQL est à la fois l'objectif de la traduction (le langage dans lequel les questions sont traduites) et un outil essentiel pour l'interaction avec les bases de données dans un projet de text-to-SQL.

3.1.2. Plateformes :

3.1.2.1. *Huggingface*



Figure 3.3: huggingface logo

Hugging-Face est une plateforme open-source et une communauté axée sur l'apprentissage automatique et le traitement du langage naturel (NLP). Fondée en 2016, elle est devenue un hub central pour le développement, le partage et l'utilisation de modèles de langage pré-entraînés et d'outils liés à l'IA.

Rôle dans notre projet text-to-SQL :

- Recherche de modèles : La plateforme nous a permis d'explorer et de découvrir divers modèles pré-entraînés pertinents pour notre tâche de text-to-SQL.
- Téléchargement de modèles : Nous avons pu facilement télécharger des modèles pré-entraînés pour les utiliser comme point de départ dans notre projet.
- Création de datasets : Hugging Face nous a offert des outils pour créer, formater et gérer nos ensembles de données, ce qui a facilité la préparation de nos données d'entraînement.
- Upload des modèles finaux : Après avoir affiné ou créé nos propres modèles, nous avons pu les partager sur la plateforme, contribuant ainsi à la communauté et rendant nos résultats accessibles à d'autres chercheurs et développeurs.
- Bibliothèque Transformers : Nous avons utilisé la bibliothèque Transformers de Hugging Face pour travailler avec les modèles de langage.

L'utilisation de Hugging Face dans notre projet a accéléré notre processus de développement, facilité l'accès aux ressources nécessaires, et nous a fourni un moyen de partager nos résultats avec la communauté de l'IA.

3.1.2.2. *Google Colab:*



Figure 3.4: Google colab logo

Google Colab, abréviation de Google Colaboratory, est un environnement de notebook Jupyter hébergé dans le cloud et gratuit. Il permet aux utilisateurs d'écrire et d'exécuter du code Python directement dans un navigateur web, sans nécessiter de configuration locale.

Rôle dans notre projet Text-to-SQL :

1. Environnement de développement : Colab nous a fourni un espace de travail préconfiguré avec les bibliothèques Python essentielles pour le machine Learning et le NLP, nous permettant de commencer rapidement notre développement.
2. Accès à des ressources de calcul : Colab nous a offert un accès gratuit à des GPU et TPU, ce qui a considérablement accéléré l'entraînement et le fine-tuning de nos modèles text-to-SQL.
3. Collaboration facile : La nature basée sur le cloud de Colab a simplifié le partage de code et de résultats au sein de notre équipe.
4. Intégration avec Google Drive : Cela nous a permis de stocker et d'accéder facilement à nos données et modèles.
5. Expérimentation rapide : L'environnement de notebook nous a facilité l'itération rapide et l'expérimentation avec différentes approches et hyperparamètres.
6. Visualisation des résultats : Colab intègre des outils de visualisation qui nous ont été utiles pour analyser les performances de nos modèles.
7. Persistance des sessions : Bien que les sessions Colab ne soient pas permanentes, elles nous ont offert une durée suffisante pour mener à bien nos tâches d'entraînement et d'évaluation.

En utilisant Colab, nous avons bénéficié d'un environnement de développement flexible et puissant, particulièrement adapté à notre projet d'apprentissage automatique, sans avoir à gérer une infrastructure locale complexe.

3.1.2.3. *Papers with code* :



Figure 3.5: papers with code logo

Papers with Code est une plateforme open-source qui rassemble des articles de recherche en apprentissage automatique avec leurs implémentations de code correspondantes. Lancée en 2019, elle vise à rendre la recherche en IA plus accessible et reproductible.

Rôle dans notre projet text-to-SQL :

1. Exploration de l'état de l'art : Cette plateforme nous a permis d'explorer les dernières avancées dans le domaine du text-to-SQL, en nous donnant accès à la fois aux articles de recherche et à leurs implémentations.
2. Benchmarks et évaluations : Nous avons pu consulter les classements des différents modèles sur des tâches spécifiques de text-to-SQL, ce qui nous a aidés à situer notre travail dans le contexte de la recherche actuelle.
3. Inspiration pour l'implémentation : Les codes sources associés aux articles nous ont fourni des idées précieuses pour l'implémentation de nos propres modèles et techniques.
4. Reproductibilité : En suivant les implémentations fournies, nous avons pu reproduire certains résultats de l'état de l'art, ce qui a servi de base solide à notre propre travail.
5. Veille technologique : La plateforme nous a permis de rester à jour sur les dernières innovations dans notre domaine, guidant ainsi nos choix méthodologiques.
6. Comparaison des performances : Nous avons pu comparer directement les performances de nos modèles avec celles publiées sur la plateforme, nous donnant un point de référence clair.

L'utilisation de Papers with Code a considérablement enrichi notre projet en nous offrant un accès facile aux dernières avancées de la recherche en text-to-SQL, tout en nous fournissant des ressources pratiques pour l'implémentation et l'évaluation de nos modèles.

3.1.3. Bibliothèques

- **Bitsandbytes:**

Définition : Une bibliothèque légère pour le calcul efficace avec des modèles de grande taille.

Utilisation : Utilisée pour quantifier et optimiser les grands modèles linguistiques afin de fonctionner efficacement sur du matériel grand public.

- **Datasets :**

Définition : Une bibliothèque de Hugging Face pour la gestion facile et efficace des ensembles de données.

Utilisation : Fournit un large éventail de jeux de données pour l'entraînement et l'évaluation des modèles de machine learning, y compris des méthodes pour prétraiter et manipuler les données.

- **Accelerate**

Définition : Une bibliothèque pour exécuter facilement des modèles PyTorch sur plusieurs appareils (CPU/GPU) avec des modifications minimales du code.

Utilisation : Simplifie le processus d'entraînement des modèles dans des environnements distribués, supportant la précision mixte et les configurations multi-GPU.

- **Loralib :**

Définition : Une bibliothèque pour l'adaptation efficace à faible rang des modèles linguistiques.

Utilisation : Utilisée pour adapter les grands modèles linguistiques à des tâches spécifiques avec moins de paramètres et de ressources.

- **Transformers :**

Définition : Une bibliothèque de Hugging Face qui fournit des modèles de machine learning à la pointe de la technologie, en particulier les transformers.

Utilisation : Largement utilisée pour implémenter et affiner les modèles transformers pour des tâches telles que la classification de texte, la génération, et plus encore.

- **Peft :**

Définition : Bibliothèque d'affinement efficace des paramètres de Hugging Face.

Utilisation : Facilite l'affinement des grands modèles en réduisant le nombre de paramètres entraînaibles, rendant le processus plus efficace.

- **Wandb :**

Définition : Weights and Biases, un outil de suivi des expériences, de surveillance des modèles et de collaboration.

Utilisation : Utilisé pour suivre et visualiser le processus d'entraînement des modèles de machine learning, enregistrer des métriques, et collaborer avec d'autres.

- **Scipy :**

Définition : Une bibliothèque Python utilisée pour le calcul scientifique et technique.

Utilisation : Fournit des modules pour l'optimisation, l'intégration, l'interpolation, les problèmes de valeurs propres, les équations algébriques, et plus encore.

- **Unsloth :**

Définition : Une bibliothèque pour l'entraînement et l'inférence rapides des modèles linguistiques.

Utilisation : Conçue pour optimiser l'entraînement et l'inférence des modèles linguistiques, en particulier sur du matériel moderne.

- **Xformers :**

Définition : Une bibliothèque pour optimiser et accélérer les transformers, y compris Flash Attention.

Utilisation : Améliore l'efficacité des modèles transformers, en particulier pour une utilisation sur GPU.

- **Torch :**

Définition : PyTorch, une bibliothèque open-source de machine learning.

Utilisation : Fournit des outils pour l'apprentissage profond et le calcul tensoriel, largement utilisée pour l'entraînement des réseaux neuronaux.

- **Tensorboard :**

Définition : Un outil pour visualiser les expériences de machine learning.

Utilisation : Utilisé pour enregistrer et visualiser des métriques pendant le processus d'entraînement, aidant à déboguer et optimiser les modèles.

- **Evaluate :**

Définition : Une bibliothèque de Hugging-Face pour évaluer les modèles de machine learning.

Utilisation : Fournit des méthodes standardisées pour évaluer la performance des modèles sur diverses tâches et benchmarks.

- **flash-attn :**

Définition : Flash Attention, un mécanisme d'attention optimisé pour les transformers.

Utilisation : Améliore l'efficacité et la vitesse des calculs d'attention dans les modèles transformers, en particulier sur les GPU modernes.

- **Diffusers :**

Définition : Une bibliothèque pour les modèles de diffusion.

Utilisation : Implémente des modèles génératifs basés sur la diffusion pour des tâches telles que la synthèse d'images et plus encore.

- **Safetensors :**

Définition : Une bibliothèque pour la gestion sécurisée des tenseurs.

Utilisation : Assure la sérialisation et la désérialisation sécurisées et efficaces des données de tenseurs.

- **huggingface_hub :**

Définition : Une bibliothèque pour accéder aux et déposer des modèles, ensembles de données et plus depuis et sur le Hugging Face Hub.

Utilisation : Fournit des outils pour interagir avec le Hugging Face Hub, permettant de partager et récupérer facilement des modèles et des ensembles de données.

- **Langchain :**

Définition : Une bibliothèque pour construire des applications avec de grands modèles linguistiques.

Utilisation : Supporte le développement d'applications complexes utilisant des modèles linguistiques, y compris l'intégration et l'orchestration de divers composants.

3.2. Systèmes de gestion de base de données

3.2.1. Postgres



Figure 3.6 : logo Postgres

PostgreSQL est un système de gestion de base de données relationnelle orienté objet puissant et open source, capable de gérer en toute sécurité des charges de travail de données complexes. Voici quelques fonctionnalités clés de PostgreSQL :

- **Système de gestion de base de données relationnelle orienté objet** : PostgreSQL combine les avantages des bases de données relationnelles avec les flexibilités des objets, permettant une manipulation de données plus sophistiquée et une meilleure intégration avec les langages de programmation modernes.
- **Open Source** : PostgreSQL est gratuit et sa source est accessible à tous, ce qui favorise son adoption et son développement continu par une communauté internationale.
- **Extensibilité** : Son fonctionnement basé sur des catalogues permet une grande extensibilité, permettant aux utilisateurs de définir leurs propres types de données, fonctions, et même d'écrire du code dans divers langages de programmation sans avoir besoin de recompiler la base de données.
- **Sécurité** : PostgreSQL offre une authentification robuste, un système de contrôle d'accès, et la possibilité de sécuriser à différents niveaux.

3.2.2. MongoDB



Figure 3.7: logo Mongo DB

MongoDB est un système de gestion de base de données NoSQL open source, conçu pour stocker des données de manière flexible et évolutive. Voici quelques fonctionnalités clés de MongoDB :

- **Base de données NoSQL orientée documents** : MongoDB utilise un format de stockage basé sur des documents BSON (Binary JSON), ce qui permet une flexibilité accrue par rapport aux bases de données relationnelles traditionnelles. Les données sont stockées sous forme de documents similaires à des objets JSON, facilitant la manipulation des données hiérarchiques et complexes.

- **Scalabilité** : MongoDB est conçu pour être facilement évolutif. Il supporte le sharding, une méthode de partitionnement des données sur plusieurs machines, permettant de gérer des charges de travail importantes et des volumes de données massifs.
- **Performances** : MongoDB est optimisé pour les performances avec des fonctionnalités comme les index, les agrégations, et le stockage en mémoire. Il peut gérer des opérations en temps réel tout en offrant des temps de réponse rapides.
- **Flexibilité du schéma** : Contrairement aux bases de données relationnelles, MongoDB permet un schéma flexible où les documents dans une collection peuvent avoir des structures de données différentes. Cela facilite les changements rapides dans les structures de données sans nécessiter de migrations complexes.
- **Sécurité** : MongoDB offre des fonctionnalités de sécurité robustes, y compris l'authentification, le contrôle d'accès basé sur les rôles (RBAC), et le chiffrement des données au repos et en transit. Il permet également l'intégration avec des systèmes d'authentification externes comme LDAP et Kerberos.
- **Haute disponibilité** : MongoDB supporte la réplication automatique des données via les ensembles de réplicas, assurant une haute disponibilité et une tolérance aux pannes.

Dans le cadre de mon projet, MongoDB a été utilisé pour stocker certains ensembles de données. Grâce à sa flexibilité et sa capacité de manipulation de données complexes, il a facilité le stockage et la récupération des informations nécessaires pour les différentes étapes de l'entraînement et de l'évaluation des modèles de langage.

3.2.3. Hugging-Face datasets



Figure 3.8:hugging face datasets logo

Hugging-Face Datasets est une bibliothèque open source conçue pour simplifier le processus de manipulation des ensembles de données dans les projets de machine Learning. Voici quelques fonctionnalités clés de Hugging-Face Datasets :

- **Large collection d'ensembles de données** : La bibliothèque offre un accès à une vaste collection d'ensembles de données pour des tâches variées telles que la classification de texte, la traduction, la génération de texte, et bien d'autres.
- **Facilité de téléchargement et de chargement** : Hugging Face Datasets permet de télécharger et de charger facilement des ensembles de données à partir de la Hugging Face Hub ou d'autres sources, avec des méthodes simples et intuitives.
- **Prise en charge des grands ensembles de données** : La bibliothèque est optimisée pour manipuler de grands ensembles de données qui ne tiennent pas en mémoire, grâce à une gestion efficace des E/S et des structures de données mémoire-mappées.
- **Interopérabilité** : Les ensembles de données peuvent être utilisés directement avec des bibliothèques de Deep Learning populaires comme PyTorch, TensorFlow, et JAX, facilitant ainsi l'intégration dans les pipelines d'entraînement de modèles.
- **Prétraitement et transformation** : Hugging Face Datasets offre des outils pour prétraiter et transformer les données, y compris la tokenisation, le filtrage,

l'échantillonnage, et l'augmentation des données, ce qui simplifie la préparation des données pour l'entraînement des modèles.

- **Interopérabilité avec d'autres bibliothèques de Hugging Face** : La bibliothèque s'intègre parfaitement avec d'autres outils de l'écosystème Hugging Face, tels que Transformers et Tokenizers, permettant une manipulation et un traitement des données cohérents et efficaces.

Dans le cadre de mon projet, Hugging Face Datasets a été utilisé pour gérer les ensembles de données nécessaires à l'entraînement et à l'évaluation des modèles. Grâce à ses capacités de manipulation de données efficaces et à son intégration avec les autres outils de Hugging Face, il a facilité le prétraitement des données et leur préparation pour les différentes étapes du projet.

3.3. Outils de communications et de réunions

3.3.1. Google Meet



Figure 3.9: logo Google Meet

Google Meet est un outil de conférence vidéo en ligne développé par Google, qui a remplacé Google Hangouts pour les appels vidéo. Il est intégré avec Google Calendar, permettant aux utilisateurs de planifier et de rejoindre des réunions vidéo directement à partir de leurs calendriers. Google Meet offre une expérience de conférence vidéo sécurisée, avec tous les flux vidéo et audio cryptés, assurant une connexion sécurisée même lorsque les participants sont hors site. Voici quelques fonctionnalités clés de Google Meet :

- **Conférences Vidéo** : Google Meet permet de réaliser des appels vidéo en temps réel, offrant une qualité vidéo nette et une latence faible.
- **Participation Sécurisée** : Toutes les communications vidéo et audio dans Google Meet sont chiffrées, protégeant la confidentialité des discussions.
- **Compatibilité Multiplate** : Les utilisateurs peuvent rejoindre des réunions Google Meet depuis n'importe quel appareil, que ce soit un smartphone, une tablette ou un ordinateur, sans avoir besoin d'installer de logiciel supplémentaire.
- **Intégration avec Google Workspace** : Google Meet est intégré à Google Workspace, permettant une collaboration en contexte directement depuis Google Docs, Sheets et Slides.
- **Support Hardware** : Google propose une gamme complète de matériel certifié Google Meet, comprenant des caméras, des microphones et des écrans interactifs, pour améliorer l'expérience de conférence dans les salles de réunion.

3.3.2. Google chat



Figure 3.10: logo Google Chat

Google Chat est une plateforme de messagerie et de collaboration en temps réel qui fait partie de Google Workspace. Elle permet aux équipes de communiquer efficacement, partager des fichiers, et collaborer sur des projets. Google Chat est conçu pour être accessible et flexible, disponible sur différents appareils, y compris les ordinateurs de bureau, les smartphones, et les tablettes. Il offre une intégration étroite avec d'autres outils de Google Workspace, tels que Google Drive, Google Calendar, et Google Meet, facilitant ainsi la productivité et la collaboration. Voici quelques fonctionnalités clés de Google Chat :

- **Messages Directs et Groupes** : Google Chat permet d'envoyer des messages directs à n'importe quelle personne ayant une adresse Gmail ou un compte Google, ainsi que de configurer des chats de groupe avec plusieurs participants.
- **Espaces** : Les espaces sont une caractéristique unique de Google Chat qui favorise la collaboration d'équipe persistante. Ils permettent de partager des fichiers, d'attribuer des tâches, et de consulter l'historique des messages. Les espaces sont idéaux pour les groupes souhaitant travailler sur un projet ou communiquer autour d'un intérêt partagé.
- **Intégrations** : Outre les intégrations prises en charge par défaut avec Gmail, Google Forms, Google Calendar, Google Meet, Google Drive, et Google Docs, Google Chat s'intègre également avec des solutions tierces populaires grâce à des API REST. Cela permet aux entreprises de personnaliser leur espace de travail selon leurs besoins spécifiques.
- **Sécurité** : Google Chat bénéficie de la même technologie anti-spam, anti-phishing, et anti-malware qui protège plus de 2 milliards de boîtes de réception Gmail. Il offre une communication sécurisée, avec des flux vidéo et audio cryptés, protégeant ainsi la confidentialité des discussions.

3.4. Outils d'industrialisation

3.4.1. Docker



Figure 3.11: logo Docker

Docker est une plateforme de conteneurisation qui permet aux développeurs de packager et de déployer des applications dans des conteneurs logiciels isolés. Ces conteneurs assurent que les applications fonctionnent uniformément dans différents environnements, ce qui facilite le développement, le déploiement et la maintenance des applications

Docker permet de conditionner et d'exécuter des applications dans des environnements isolés nommés conteneurs. Ces conteneurs assurent une isolation et une sécurité élevées, permettant à plusieurs conteneurs de fonctionner conjointement sur un seul hôte. Ils sont autonomes et intègrent tous les éléments requis pour exécuter l'application, supprimant ainsi la dépendance à l'égard de la configuration spécifique de l'hôte. Dans notre projet, tous les outils implémentés sont basés sur ces conteneurs.

- **Conteneurisation** : Docker utilise des conteneurs pour emballer les applications et leurs dépendances dans un seul package exécutable. Ce package contient tout ce dont l'application a besoin pour fonctionner, y compris le code, les bibliothèques, les variables d'environnement et les outils système.
- **Isolation** : Chaque conteneur est isolé des autres, ce qui signifie qu'il peut s'exécuter sur n'importe quel système d'exploitation sans interférer avec les autres conteneurs ou avec le système hôte.
- **Portabilité** : Les conteneurs Docker peuvent s'exécuter sur n'importe quelle machine qui a Docker installé, ce qui facilite le déploiement et la mise à l'échelle des applications.

3.4.1.1. Fonctionnement de Docker

- **Images Docker** : Les images Docker sont des modèles légers qui contiennent les instructions pour construire un conteneur. Elles servent de base pour créer des conteneurs personnalisés.
- **Conteneurs Docker** : Les conteneurs sont des instances d'images Docker qui peuvent être lancées et arrêtées. Ils encapsulent l'application et son environnement.
- **Registres Docker** : Les registres Docker sont des dépôts où les images Docker sont stockées et partagées. Docker Hub est le registre officiel, mais il existe aussi d'autres registres privés.
- **Dockerfile** : Le Dockerfile est un fichier texte qui contient les instructions pour construire une image Docker. Il spécifie les étapes nécessaires pour créer l'image, y compris l'utilisation des images de base, l'installation des dépendances, et la configuration de l'application.

3.4.1.2. Avantages de Docker

- **Développement rapide** : Les développeurs peuvent construire, tester et déployer des applications plus rapidement grâce à l'isolation et à la portabilité des conteneurs.
- **Consommation de ressources** : Les conteneurs sont plus légers que les machines virtuelles, ce qui signifie qu'ils consomment moins de ressources système.
- **Scalabilité** : Docker facilite la mise à l'échelle des applications en permettant de démarrer rapidement de nouveaux conteneurs pour gérer les charges de travail.
- **Automatisation** : Docker s'intègre avec des outils d'intégration continue et de déploiement continu, permettant une automatisation complète du pipeline de déploiement.

Docker est donc une technologie essentielle pour les développeurs et les opérations, offrant une solution robuste et flexible pour le développement, le déploiement et la gestion des applications.

3.4.2. Git



Figure 3.12: logo Git

Git est un système de contrôle de version distribué, libre et open source créée par Linus Torvalds en 2005. Il est devenu l'un des outils de gestion de code source les plus populaires et largement utilisé dans le développement de logiciels et de projets informatiques. Voici quelques-unes de ses fonctionnalités :

- **Collaboration** : Git est conçu pour faciliter la collaboration entre les développeurs. Il offre des outils pour partager des versions de code, gérer les conflits et gérer les autorisations d'accès.
- **Distribution** : Git est un système de contrôle de version distribué, ce qui signifie que toutes les copies de l'historique des versions sont stockées localement.
- **Branchement et fusionnement** : Git permet aux développeurs de travailler sur des fonctionnalités ou des corrections de bugs en créant des branches distinctes. Les modifications peuvent ensuite être fusionnées avec la branche principale en utilisant un processus de fusion.
- **Révisions** : les commits sont stockés sous forme de révisions, chacune contenant un ensemble de modifications de code. Git offre la possibilité d'explorer l'historique des versions et restaurer les révisions précédentes.

En ce qui concerne le flux de Git, nous présentons le schéma ci-dessous qui décrit l'ensemble des phases utilisées par Git pour envoyer les changements vers et les recevoir depuis le repository distant :

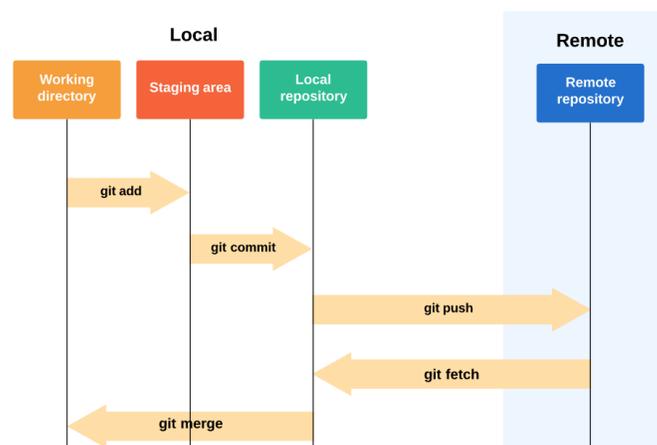


Figure 3.13: flux Git

Tout au long du flux Git, les développeurs apportent des modifications dans le répertoire de travail, les déplacent vers la zone de staging à l'aide de « **git add** » pour les préparer à la validation, les valident dans le référentiel local avec « **git commit** », puis poussent ces validations vers le repository distant à l'aide de « **git push** » pour les mettre à disposition des collaborateurs.

Pour extraire les modifications du repository distant vers le répertoire de travail, nous utilisons d'abord « **git fetch** », qui apporte les modifications du référentiel distant vers le référentiel local. Ensuite, nous utilisons « **git merge** » pour fusionner les modifications du référentiel local dans le répertoire de travail. Cela garantit que les dernières modifications du référentiel distant sont incorporées dans le répertoire de travail, ce qui permet la synchronisation.

3.4.3. GitHub



Figure 3.14: logo GitHub

GitHub est une plateforme de développement collaboratif qui permet aux développeurs de travailler ensemble sur des projets logiciels. Elle offre des fonctionnalités pour la gestion de code, le suivi des bugs, la planification des projets, et bien plus encore. Il se caractérise par :

- **Gestion de Code Centralisée** : GitHub permet aux développeurs de suivre les modifications de code, de collaborer sur des projets, et de gérer des versions de manière efficace. Les utilisateurs peuvent cloner des dépôts, soumettre des modifications, et fusionner des demandes de tirage.
- **Collaboration Facilitée** : La plateforme offre des outils pour la collaboration en temps réel, y compris des discussions intégrées, des revues de code, et des notifications pour garder tout le monde informé des dernières mises à jour.
- **Intégration Continue et Déploiement Continu** : GitHub s'intègre avec des outils d'intégration continue et de déploiement continu, permettant aux équipes de tester et de déployer leur code automatiquement.
- **Communauté et Apprendre** : GitHub héberge une communauté active de développeurs qui partagent des connaissances, prodiguent des conseils, et contribuent à des projets open source.
- **Sécurité et Confidentialité** : GitHub offre des options de sécurité robustes, y compris l'authentification à deux facteurs, des permissions granulaires pour le code, et la possibilité de rendre des dépôts privés.

3.4.4. PyCharm



Figure 3.15: logo de PyCharm IDEA

PyCharm est un environnement de développement intégré (IDE) créé par JetBrains, spécifiquement conçu pour le développement en Python. Il est reconnu pour sa puissance et sa popularité parmi les développeurs Python, offrant une multitude d'outils et de fonctionnalités pour améliorer la productivité.

Voici quelques fonctionnalités clés de PyCharm qui ont été utiles durant le projet :

- **Assistance et analyse du code** : PyCharm fournit une assistance avancée pour le code Python, incluant des suggestions, des correctifs rapides et des importations automatiques. Il effectue une analyse statique du code pour détecter les erreurs, mettre en évidence les problèmes potentiels et proposer des suggestions pour améliorer la qualité du code.
- **Intégration du contrôle de version** : PyCharm s'intègre de manière transparente aux systèmes de contrôle de version populaires tels que Git, Subversion, Mercurial et autres. Il offre des outils visuels de comparaison et de fusion, la gestion des branches et une navigation facile dans l'historique des commits.
- **Outils de construction et de déploiement** : PyCharm prend en charge divers outils de construction et systèmes de gestion des dépendances tels que pip, setuptools et Poetry. Il facilite la création, l'exécution, le test et le déploiement d'applications avec des configurations d'exécution/débogage configurables.
- **Outils de système de base de données** : PyCharm propose une intégration profonde avec les systèmes de gestion de bases de données (DBMS) populaires, offrant une expérience de développement optimisée pour les développeurs travaillant avec des bases de données. Parmi les fonctionnalités clés, on trouve :
 - **Support pour multiples DBMS** : L'IDE prend en charge une variété de systèmes de gestion de bases de données, y compris MySQL, PostgreSQL, Oracle, SQL Server, et MongoDB, permettant aux développeurs de travailler avec leurs choix de base de données préférés sans avoir besoin de sortir de l'environnement de développement.
 - **Génération de requêtes SQL assistée** : PyCharm offre des fonctionnalités d'autocomplétion et de suggestion pour la génération de requêtes SQL, aidant les développeurs à écrire rapidement et avec précision des requêtes complexes.
 - **Visualisation des données** : Les développeurs peuvent visualiser les données directement dans l'IDE, ce qui facilite la compréhension des structures de données et l'identification des relations entre différentes tables ou collections.
 - **Migration de bases de données** : L'IDE inclut des outils pour migrer des schémas de bases de données entre différents DBMS, simplifiant le processus de migration et d'intégration de nouvelles bases de données.

Dans le cadre de mon projet, PyCharm a été utilisé pour développer et gérer les scripts Python nécessaires à l'intégration des modèles de langage dans le projet OneGate.

Grâce à ses fonctionnalités avancées d'assistance au codage, d'intégration des bases de données et de gestion des versions, il a permis de rationaliser le processus de développement et d'assurer une gestion efficace des différentes étapes du projet.

Conclusion

Dans ce chapitre, nous avons présenté l'ensemble des technologies utilisés dans afin d'accomplir le travail demandé. Le chapitre suivant entamera l'étude des besoins fonctionnels ainsi que la conception du système qui détaillera les différentes interactions entre les entités de systèmes.

Chapitre 4. Réalisation et concrétisation

Introduction

Ce chapitre présentera la dernière étape de la mise en œuvre de la solution, après avoir présenté le contexte général du projet et sa conception dans les chapitres précédents, ce chapitre a pour but de concrétiser ce qui est décrit théoriquement. La première partie est dédiée à la description du processus et méthodologie suivis, la deuxième partie est consacrée pour présenter l'implémentation et finalement la description du résultat du travail en mettant en évidence des captures d'écran pour la démonstration.

4.1. Démarche suivie

Dans cette section, nous décrivons en détail le processus et la méthodologie suivis pour mettre en œuvre la solution de fine-tuning des modèles de langage.

Ce processus comprend plusieurs étapes clés, allant de la collecte et du nettoyage des données à la sélection des modèles et des méthodes de fine-tuning, ainsi que l'évaluation des performances du système. Chacune de ces étapes est cruciale pour assurer la robustesse et l'efficacité du modèle final, capable de générer des requêtes SQL précises et de s'adapter à différents domaines d'application. Nous présentons également les défis rencontrés et les solutions adoptées pour les surmonter, mettant en lumière les meilleures pratiques et les leçons apprises au cours du projet.

4.2. Collection des données pour fine-tuning

Nous avons collecté des données à partir de diverses sources fournissant des données au format schéma-question-sql et nous les avons regroupées en un seul ensemble de données.

Voici la distribution des données :

Source de données	Nombre d'entrées
gretelai_synthetic_text_to_sql	100000
wikisql	80442
sql_create_context	77111
nvbench	22968
css	22050
mimicsql_data	19987
squall	10595
mimic_iii	10078
sede	10047
eicu	10025
bird	9428
spider	6081
atis	4895
advising	4381
criteria2sql	1997
scholar	776
academic	185
Imdb et yelp	259

Table 2: distribution des données dans le dataset

Pour le nettoyage de ces données Nous avons nettoyé les données en conservant autant de domaines que possible, car l'utilisation de notre système est prévue pour tous types de domaines (éducation, finance, santé, etc.).

4.3. Choix du model

Nous avons effectué des recherches approfondies pour sélectionner les modèles les plus appropriés pour notre projet. Après avoir évalué plusieurs options, nous avons réduit notre choix à trois modèles : LLama, Mistral et Gemma. Voici les critères qui ont guidé notre sélection :

- **Taille du modèle :**

- **LLama** : Disponible en plusieurs tailles, allant de 7 milliards à 65 milliards de paramètres, LLama offre une flexibilité en termes de capacité de traitement en fonction des besoins spécifiques du projet.
- **Mistral** : Connu pour ses modèles efficaces, Mistral propose des modèles de 7 milliards de paramètres, ainsi qu'une variante "mixture-of-experts" de 8x7B paramètres, permettant des performances optimisées avec une taille relativement compacte.
- **Gemma** : Gemma est disponible en tailles de 2 milliards et 7 milliards de paramètres. Conçus pour être efficaces en termes de ressources, ces modèles sont adaptés aux appareils avec des contraintes de calcul.

- **Vitesse :**

La rapidité d'exécution est cruciale pour assurer une expérience utilisateur fluide. Nous avons évalué chaque modèle sur la base de leur temps de réponse lors de l'exécution de requêtes SQL. Mistral a montré des temps de réponse très rapides grâce à son architecture optimisée. LLama et Gemma ont également offert de bonnes performances, bien que légèrement plus lentes que Mistral en termes de latence.

- **Performance :**

Nous avons mesuré la précision des réponses générées par chaque modèle en exécutant des requêtes SQL sur des bases de données variées.

Gemma a montré une excellente précision, particulièrement pour des requêtes complexes.

LLama a fourni des résultats très fiables, faisant de lui un bon choix pour une utilisation générale.

Mistral a bien performé, notamment dans des environnements nécessitant des réponses rapides avec des schémas très grandes.

- **Open Source :**

L'accessibilité et la capacité d'adaptation des modèles open source étaient des critères importants pour nous.

LLama, Mistral, et Gemma sont tous des modèles open source, ce qui nous permet de les adapter à nos besoins spécifiques.

- **Facilité d'intégration :**

Nous avons également considéré la facilité avec laquelle chaque modèle pouvait être intégré dans notre pipeline existant.

LLama et Mistral sont connus pour leur documentation complète et leur compatibilité avec de nombreux outils de développement, facilitant leur intégration.

Gemma offre également une bonne intégration, bien que nécessitant parfois des ajustements supplémentaires pour des configurations spécifiques.

En résumé, notre choix s'est porté sur LLama, Mistral et Gemma en raison de leur équilibre entre taille, vitesse, performance, et leur nature open source. Ces modèles répondaient le mieux à nos besoins spécifiques, notamment en termes de précision des requêtes SQL, de rapidité de réponse, et de flexibilité pour diverses applications et domaines.

4.4. Choix de la méthode du fine-tuning

- **C'est Quoi Fine-tuning**

Le fine-tuning (ou affinage en français) est une technique d'apprentissage par transfert qui consiste à prendre un modèle pré-entraîné sur une grande quantité de données générales et à l'adapter à une tâche spécifique ou à un domaine particulier. Cette approche permet d'obtenir de meilleures performances sur des tâches spécialisées tout en utilisant moins de données et de ressources de calcul que si l'on devait entraîner un modèle à partir de zéro.

- **Exemple :**

Imaginons que nous ayons un modèle de langue pré-entraîné sur un large corpus de textes généraux. Si nous voulons l'utiliser pour analyser des critiques de films, nous pouvons le fine-tuner sur un ensemble de données de critiques cinématographiques. Cela permettra au modèle d'apprendre le vocabulaire spécifique au cinéma, les nuances de l'analyse filmique, et d'améliorer ses performances dans ce domaine particulier.

- **Types de Fine-tuning :**

Il existe plusieurs types de fine-tuning :

1. **Fine-tuning complet** : Tous les paramètres du modèle sont ajustés pendant l'entraînement. C'est la méthode la plus coûteuse en termes de calcul et de mémoire.
2. **Fine-tuning des dernières couches** : Seules les dernières couches du modèle sont ajustées, tandis que les premières couches restent figées. Cette méthode est moins coûteuse et peut être efficace pour de nombreuses tâches.
3. **Fine-tuning avec ajout de couches** : De nouvelles couches sont ajoutées au modèle pré-entraîné et seules ces nouvelles couches sont entraînées.
4. **Fine-tuning efficace en paramètres** : C'est ici qu'interviennent les méthodes que nous avons choisies. Ces approches visent à réduire le nombre de paramètres ajustés tout en maintenant les performances.

- **Parameter efficient fine tuning :**

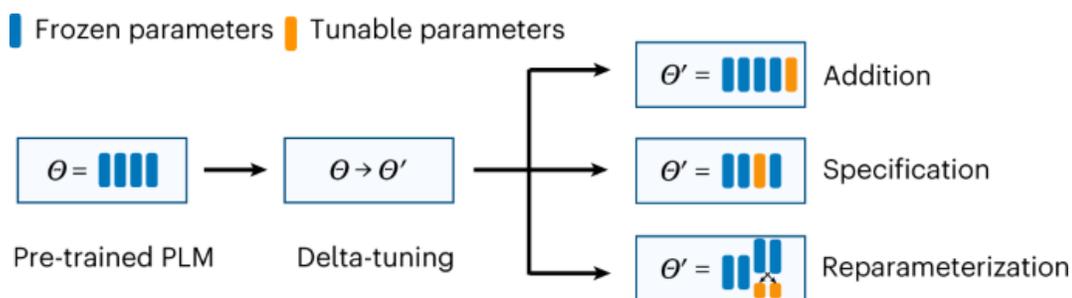


Figure 4.1 parameter efficient fine tuning

Les approches PEFT n'affinent qu'un petit nombre de paramètres (supplémentaires) du modèle tout en gelant la plupart des paramètres des LLM pré-entraînés, réduisant ainsi considérablement les coûts de calcul et de stockage. Cela permet également de surmonter les problèmes d'**oubli catastrophique**, un comportement observé lors du fine-tuning complet des LLM. Les approches PEFT se sont également révélées meilleures que le fine-tuning dans les régimes à faibles données et se généralisent mieux aux scénarios hors domaine.

Cela facilite également la portabilité, où les utilisateurs peuvent ajuster les modèles en utilisant les méthodes PEFT pour obtenir de minuscules points de contrôle de quelques Mo, comparés aux grands points de contrôle du fine-tuning complet. Par exemple, `bigscience/mt0-xxl` occupe 40 Go de stockage et un fine-tuning complet conduirait à des points de contrôle de 40 Go pour chaque jeu de données en aval, alors qu'en utilisant les méthodes PEFT, il ne s'agirait que de quelques Mo pour chaque jeu de données en aval, tout en obtenant des performances comparables au fine-tuning complet. Les petits poids entraînés par les approches PEFT sont ajoutés au-dessus du LLM pré-entraîné. Ainsi, le même LLM peut être utilisé pour plusieurs tâches en ajoutant de petits poids sans avoir à remplacer l'ensemble du modèle.

En résumé, les approches PEFT vous permettent d'obtenir des performances comparables au fine-tuning complet tout en n'ayant qu'un petit nombre de paramètres entraînaibles.

Parlons maintenant des trois méthodes spécifiques que nous avons utilisées :

- PEFT (Parameter-Efficient Fine-Tuning) est une approche générale qui englobe plusieurs techniques visant à réduire le nombre de paramètres ajustés lors du fine-tuning. L'idée principale est de ne modifier qu'une petite partie des paramètres du modèle, ce qui permet d'économiser des ressources de calcul et de mémoire tout en obtenant des performances comparables à celles d'un fine-tuning complet.
- LoRA (Low-Rank Adaptation) est une technique spécifique de PEFT. Elle fonctionne en ajoutant des matrices de faible rang aux poids existants du modèle. Ces matrices sont les seuls paramètres ajustés pendant le fine-tuning. L'avantage de LoRA est qu'il permet une adaptation efficace du modèle avec un nombre très réduit de paramètres entraînaibles, tout en préservant la capacité du modèle à généraliser.
- QLoRA (Quantized Low-Rank Adaptation) va encore plus loin en combinant LoRA avec des techniques de quantification. La quantification consiste à réduire la précision numérique des paramètres du modèle (par exemple, en passant de nombres à virgule flottante 32 bits à des entiers 8 bits). QLoRA applique cette quantification au modèle de base tout en utilisant LoRA pour le fine-tuning. Cela permet une utilisation encore plus efficace de la mémoire, rendant possible le fine-tuning de grands modèles de langage sur des GPU avec une mémoire limitée, tout en maintenant une qualité comparable à celle des méthodes non quantifiées.

4.5. Code du fine-tuning.

Fine-tuning du model gemma 2B avec QLoRa :

- Installation des dépendances :

```
!pip install -q bitsandbytes datasets accelerate loralib  
!pip install -q git+https://github.com/huggingface/transformers.git@main git+https://github.com/huggingface/peft.git
```

Cette cellule installe les bibliothèques nécessaires pour le projet. bitsandbytes, datasets, accelerate, et loralib sont des outils essentiels pour l'entraînement de modèles de langage. Les versions les plus récentes de transformers et peft sont installées directement depuis GitHub.

- Connexion à Hugging-Face :

```
!huggingface-cli login
```

Cette commande permet de se connecter à Hugging Face, une plateforme populaire pour le partage de modèles et de datasets en NLP.

- Vérification du GPU :

```
!nvidia-smi -L
```

Cette commande liste les GPUs NVIDIA disponibles sur le système, ce qui est utile pour s'assurer que le matériel nécessaire est présent pour l'entraînement.

- Configuration du modèle :

```
!pip install -U transformers  
  
from google.colab import userdata  
import os  
os.environ["CUDA_VISIBLE_DEVICES"]="0"  
import torch  
import torch.nn as nn  
import bitsandbytes as bnb  
from transformers import AutoTokenizer, AutoConfig, AutoModelForCausalLM  
  
model = AutoModelForCausalLM.from_pretrained(  
    "google/gemma-2b",  
    load_in_8bit=True,  
    device_map='auto',  
)  
tokenizer = AutoTokenizer.from_pretrained("google/gemma-2b")
```

Cette section configure l'environnement et charge le modèle Gemma-2b de Google. Le modèle est chargé en 8-bit pour économiser de la mémoire GPU. Le Tokenizer correspondant est également chargé.

- Gel des poids originaux :

```

for param in model.parameters():
    param.requires_grad = False
    if param.ndim == 1:
        param.data = param.data.to(torch.float32)

model.gradient_checkpointing_enable()
model.enable_input_require_grads()

class CastOutputToFloat(nn.Sequential):
    def forward(self, x): return super().forward(x).to(torch.float32)
model.lm_head = CastOutputToFloat(model.lm_head)

```

Cette partie gère les poids originaux du modèle pour éviter de les modifier lors du fine-tuning.

Certains paramètres sont convertis en float32 pour la stabilité.

Le gradient checkpointing est activé pour réduire l'utilisation de la mémoire.

- Configuration des adaptateurs LoRA :

```

def print_trainable_parameters(model):
    trainable_params = 0, all_param = 0
    for _, param in model.named_parameters():
        all_param += param.numel()
        if param.requires_grad:
            trainable_params += param.numel()
    print(
        f"trainable params: {trainable_params} || all params: {all_param} || trainable%: {100 * trainable_params / all_param}"
    )

from peft import LoraConfig, get_peft_model

config = LoraConfig(
    r=16,
    lora_alpha=32, lora_dropout=0.05, bias="none", task_type="CAUSAL_LM"
)
model = get_peft_model(model, config)
print_trainable_parameters(model)

```

Cette section configure les adaptateurs LoRA (Low-Rank Adaptation) pour le fine-tuning efficace du modèle. LoRA permet d'ajuster le modèle avec un nombre réduit de paramètres entraînaibles.

- Préparation des données :

```

from datasets import load_dataset

system_message = """You are an text to SQL query translator. Users will ask you questions in English and you will
generate a SQL query based on the provided SCHEMA.
SCHEMA:
{schema}"""

def create_conversation(sample):
    return {
        "messages": [
            {"role": "system", "content": system_message.format(schema=sample["context"])},
            {"role": "user", "content": sample["question"]},
            {"role": "assistant", "content": sample["answer"]}
        ]
    }

dataset = load_dataset("OneGate/OGText2SQL", split="train")
dataset = dataset.shuffle().select(range(12500))
dataset = dataset.map(create_conversation, remove_columns=dataset.features, batched=False)
dataset = dataset.train_test_split(test_size=2500/12500)

print(dataset["train"][345]["messages"])

dataset["train"].to_json("train_dataset.json", orient="records")
dataset["test"].to_json("test_dataset.json", orient="records")

```

Cette partie charge et prépare le dataset pour l'entraînement. Le dataset OGText2SQL est utilisé, qui contient des paires de questions en anglais et de requêtes SQL correspondantes. Les données sont formatées pour correspondre au format d'entrée attendu par le modèle.

- Chargement et préparation d'un dataset personnalisé :

```

from datasets import Dataset, DatasetDict
import json

with open("sql_create_context_v4.json", "r") as f:
    json_data = json.load(f)

question = [item["question"] for item in json_data]
context = [item["context"] for item in json_data]
answer = [item["answer"] for item in json_data]

dataset = Dataset.from_dict({"question": question, "context": context, "answer": answer})
dataset = DatasetDict({"train": dataset})

def merge_columns(example):
    example["prediction"] = example["question"] + example["context"] + " ->: " + example["answer"]
    return example

dataset['train'] = dataset['train'].map(merge_columns)

print(dataset['train'][8])

dataset['train'][0]

dataset = dataset.map(lambda samples: tokenizer(samples['prediction']), batched=True)

```

Cette section charge un dataset personnalisé à partir d'un fichier JSON, le formate et le prépare pour l'entraînement. Les colonnes sont fusionnées pour créer une colonne "prédiction" qui sera utilisée pour l'entraînement.

- Configuration et lancement de l'entraînement :

```
import transformers
trainer = transformers.Trainer(
    model=model,
    train_dataset=dataset['train'],
    args=transformers.TrainingArguments(
        per_device_train_batch_size=4,
        gradient_accumulation_steps=4,
        warmup_steps=100,
        max_steps=50,
        learning_rate=2e-4,
        fp16=True,
        logging_steps=1,
        output_dir='outputs'
    ),
    data_collator=transformers.DataCollatorForLanguageModeling(tokenizer, mlm=False)
)
model.config.use_cache = True
trainer.train()
```

Cette partie configure et lance l'entraînement du modèle en utilisant la classe Trainer de Hugging-Face

Les paramètres d'entraînement sont définis, y compris la taille du batch, les étapes de warmup, le taux d'apprentissage, etc.

- Partage du modèle sur Hugging-Face Hub :

```
!huggingface-cli login

model.push_to_hub("MatrixIA/Gemma-2b-ft-sql",
    use_auth_token=True,
    commit_message="basic training",
    private=True)
```

Cette section permet de partager le modèle entraîné sur le Hugging-Face Hub pour qu'on ne le perd pas après la fermeture de la session. Aussi pour le partager avec la communauté de Hugging-Face.

De la même façon on a effectué le fine-tuning des autres modèles : **mistral 7b** et **Llama3 7b** Ensuite on a changé la méthode de fine-tuning à Lora et QLora et on a répété pour les trois modèles.

4.6. Evaluation des résultats de performance et rapidité

- **Comment évaluer**

Pour évaluer la performance de nos modèles fine-tunés, nous avons opté pour une approche basée sur le feedback humain, qui nous a permis d'obtenir une évaluation qualitative et précise des résultats.

Nous avons mis en place un processus d'évaluation rigoureux comprenant les étapes suivantes :

1. Préparation d'un ensemble de schémas de base de données variés.
2. Élaboration d'une série de questions diverses couvrant différents aspects des requêtes SQL.
3. Génération de requêtes SQL par chaque modèle en réponse à ces questions.
4. Test des requêtes générées sur les bases de données réelles correspondantes.
5. Évaluation manuelle de l'exactitude et de la pertinence des résultats obtenus.

Cette méthode nous a permis de comparer efficacement les performances des différents modèles que nous avons fine-tunés, notamment Llama 3 7B, Mistral 7B et Gemma 2B.

Comparison de LLaMA 3 7B, Mistral 7B, and Gemma 2B

Model	Speed	Accuracy of Query
LLaMA 3 7B finetuned	Moderate	High
Mistral 7B finetuned	Moderate	Moderate to High
Gemma 2B finetuned	Fast	Moderate

Table 3: Comparaison des LLM fine-tuned

- **Des résultats**

On va tester le model en posant des questions sur la base de donnes suivante, qui est a un schéma assez complexe :

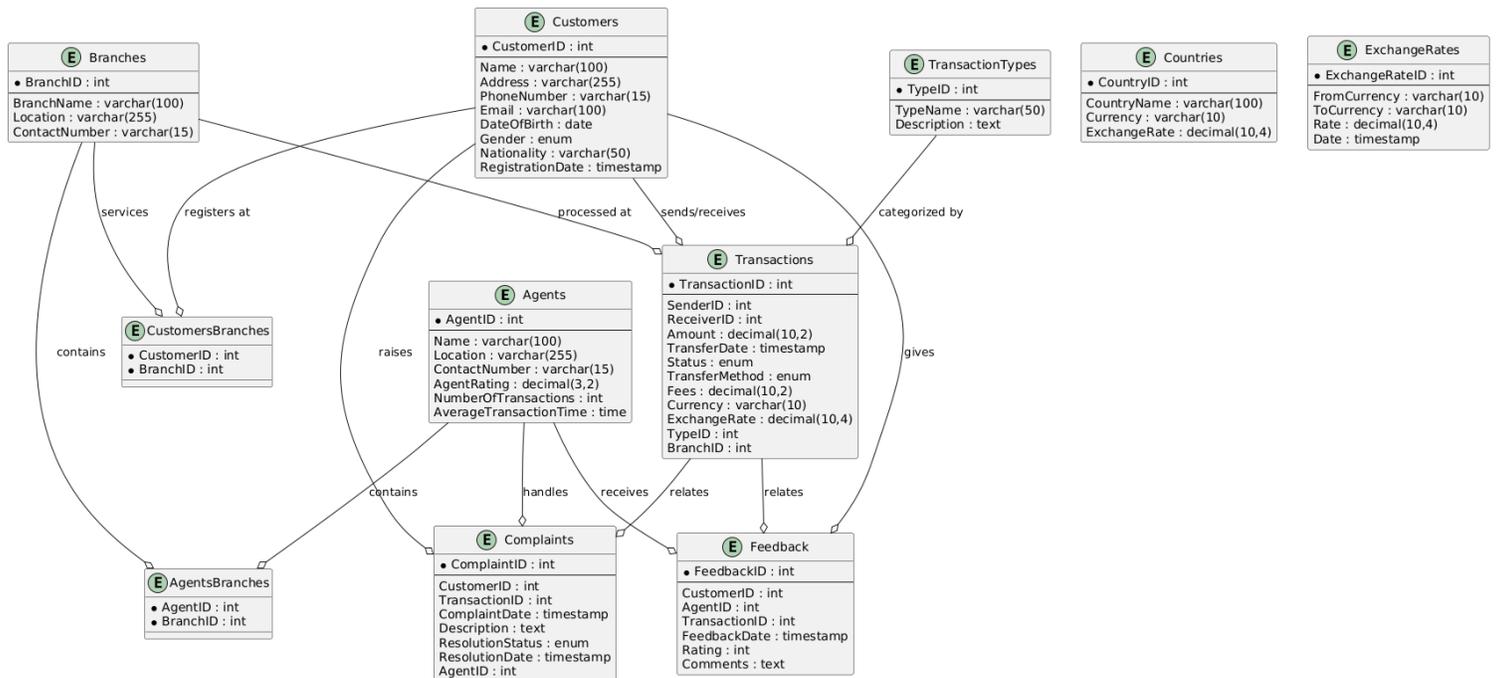


Figure 4.2 : Schéma d'une base de donnes pour une agence de transfert d'argent

Voice quelque questions en langage naturel et leurs réponses du model :

Question 1:

Is it safe to give customer with id 296 a loan?

Response du model:

```

SELECT
  CASE
    WHEN cs.Score > 300
      AND NOT EXISTS (SELECT 1 FROM Loans l WHERE l.CustomerID = c.CustomerID AND l.Status IN ('Pending', 'Rejected'))
      AND (SELECT SUM(e.MonthlyExpenditure) FROM Expenditures e WHERE e.CustomerID = c.CustomerID) < 2000
      -- Adjust this threshold based on what you consider safe
      AND NOT EXISTS (SELECT 1 FROM Payments p JOIN Loans l ON p.LoanID = l.LoanID WHERE l.CustomerID = c.CustomerID AND p.PaymentDate IS NULL)
    THEN 'Yes' ELSE 'No'
  END AS LoanApprovalStatus
FROM
  Customers c
LEFT JOIN
  CreditScores cs ON c.CustomerID = cs.CustomerID
WHERE
  c.CustomerID = 369;
  
```

Question 2:

What is the financial profile and solvency status of customer with id 288?

Response du model:

```
SELECT c.CustomerID, c.Name, c.Address, c.PhoneNumber, c.Email, c.DateOfBirth, c.Gender, c.Nationality, c.RegistrationDate,
       cs.Score, cs.LastUpdated AS CreditScoreLastUpdated, cs.ScoreSource,
       l.LoanID, l.LoanTypeID, l.CollateralID, l.AgentID, l.BranchID, l.LoanAmount,
       l.ApplicationDate, l.ApprovalDate, l.DisbursementDate, l.InterestRate, l.RepaymentTerm, l.Status,
       SUM(e.MonthlyExpenditure) AS TotalMonthlyExpenditure
FROM Customers c
LEFT JOIN CreditScores cs ON c.CustomerID = cs.CustomerID
LEFT JOIN Loans l ON c.CustomerID = l.CustomerID
LEFT JOIN Expenditures e ON c.CustomerID = e.CustomerID
WHERE c.CustomerID = 288
GROUP BY c.CustomerID, cs.Score, cs.LastUpdated, cs.ScoreSource, l.LoanID;
```

Question 3 :

*Listez les clients qui ont reçu des notifications concernant des "rendez-vous " au cours des 16 derniers mois.
?*

Response du model:

```
SELECT DISTINCT c.CustomerID, c.Name, c.Email
FROM Customers c
JOIN Notifications n ON c.CustomerID = n.CustomerID
WHERE n.NotificationDate >= DATE_SUB(CURDATE(), INTERVAL 16 MONTH)
AND n.Message LIKE '%rendez-vous%'
ORDER BY c.CustomerID;
```

4.7. Le model en action dans OneGate

Dans cette partie on présent les résultats dans la plateforme OneGate, Apres que l'utilisateur charge sa base de donnes, in peut la sélectionner et commencer à poser des questions en langage naturel :

- **Démonstration :**

Q Quelle est la répartition des statuts des transactions

The Result Breakdown

Export Send to Dashboard

Status	Count
Pending	123
Failed	122
Completed	107

< 1 >

Figure 4.3: réponse du model sur la question : *Quelle est la répartition des statuts des transactions*

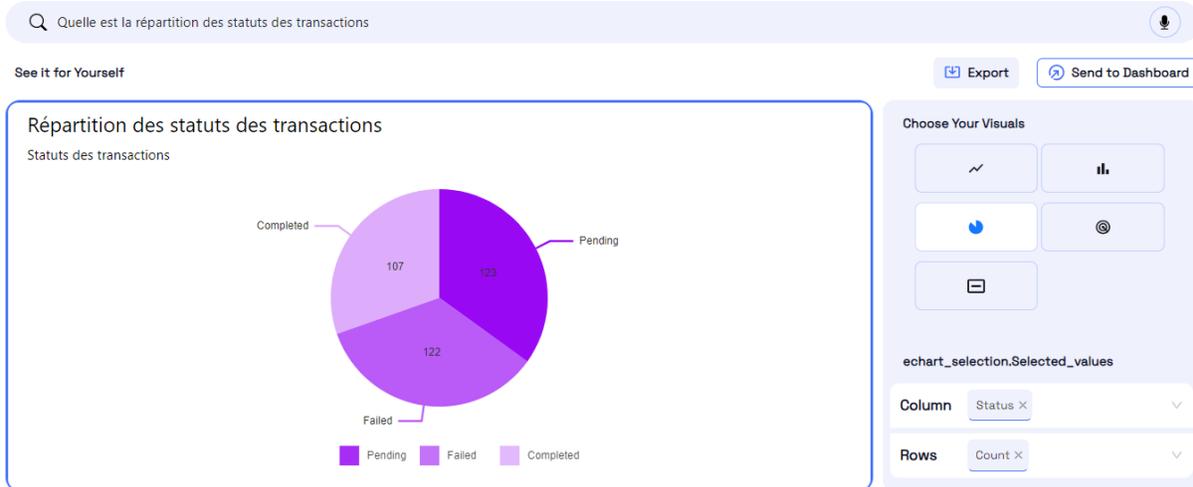


Figure 4.4: visualisation de la réponse sur la question: *: Quelle est la répartition des statuts des transactions*

🔊

The Result Breakdown Export Send to Dashboard

Name	TransactionCount
Youssef El Arabi	18
Hassan El Glaoui	18
Fatima Zahra El Mansouri	18
Meriem Bensalah	16
Anas El Ghomari	0

< 1 2 3 4 5 ... 8 >

Figure 4.5: réponse sur la question : Pouvez-vous fournir le nombre de transactions pour chaque agent en indiquant son nom et le nombre des transactions ?

Conclusion

Ce chapitre a présenté la mise en œuvre finale de notre solution, concrétisant les concepts théoriques précédemment abordés.

Nous avons détaillé notre démarche méthodologique, de la collecte des données au choix du modèle et de la méthode de fine-tuning.

Le processus de fine-tuning a été illustré par le code utilisé, suivi d'une évaluation approfondie des performances et de la rapidité du modèle.

Enfin, nous avons démontré l'application pratique de notre solution dans l'environnement OneGate, appuyant nos résultats par des captures d'écran illustratives.

Cette approche structurée a permis de présenter de manière cohérente l'ensemble du processus, de la conception à l'implémentation concrète de notre solution.

Conclusion : Lesson, Défis et Perspectives d'avenir

Notre stage au sein de Peacock s'est révélé être une expérience enrichissante et formatrice, centrée sur l'application des modèles de langage de grande taille (LLM) dans le contexte de générations des requêtes de bases de données.

Ce projet ambitieux nous a permis de plonger au cœur des technologies d'intelligence artificielle les plus avancées, tout en travaillant sur des problématiques concrètes du monde de l'entreprise.

La collaboration étroite avec une équipe pluridisciplinaire, composée de data scientists, de développeurs web et d'ingénieurs de données, a considérablement élargi notre compréhension des enjeux et des dynamiques propres aux projets d'envergure dans le domaine de l'IA.

Cette expérience nous a offert une perspective unique sur l'application pratique des LLM et les défis associés à leur intégration dans des systèmes de gestion de bases de données complexes.

Parmi les principaux défis rencontrés, la gestion des ressources computationnelles s'est avérée cruciale, particulièrement lors du fine-tuning des modèles.

Cette contrainte nous a poussés à optimiser nos approches et à faire preuve de créativité dans nos méthodologies. De plus, la gestion des attentes face aux résultats du fine-tuning a constitué un défi majeur, nous confrontant parfois à des performances décevantes malgré des investissements importants en temps et en ressources.

Ces expériences ont souligné l'importance d'une approche méthodique et itérative dans le développement de solutions basées sur l'IA, ainsi que la nécessité de développer une compréhension nuancée des facteurs influençant la qualité des résultats du fine-tuning pour les tâches de génération de requêtes.

En termes de perspectives, notre projet a mis en lumière plusieurs pistes prometteuses.

Nous attendons avec intérêt l'émergence de modèles plus compacts mais tout aussi performants, qui pourraient être fine-tunés plus efficacement tout en offrant des résultats comparables aux modèles actuels de grande taille.

Cette évolution pourrait grandement faciliter l'intégration des LLM dans un plus large éventail d'applications de bases de données.

Une direction particulièrement excitante que nous envisageons d'explorer est l'intégration d'un mécanisme d'auto-correction à notre système.

L'idée serait de permettre au LLM de recevoir les erreurs générées par le système de gestion de base de données lorsqu'une requête incorrecte est produite, et d'utiliser ces informations pour améliorer continuellement la qualité de ses générations de requêtes.

Ce stage nous a non seulement permis d'acquérir des compétences techniques précieuses dans le domaine des LLM et des bases de données, mais il nous a également offert une vision plus claire

des enjeux futurs de ce domaine en pleine expansion. Il a renforcé notre conviction que l'avenir de ce secteur réside dans le développement de systèmes plus intelligents, capables d'apprendre et de s'adapter en temps réel aux retours qu'ils reçoivent.

En conclusion, bien que les défis rencontrés aient été nombreux, ils ont représenté autant d'opportunités d'apprentissage et d'innovation. Les perspectives ouvertes par ce projet sont enthousiasmantes, promettant des avancées significatives dans la manière dont nous interagissons avec les bases de données complexes via des interfaces en langage naturel. Nous sommes impatients de continuer à explorer ces nouvelles frontières, convaincus que les leçons apprises et les idées générées au cours de ce projet contribueront de manière significative à l'évolution de ce domaine passionnant de l'IA appliquée aux bases de données.

Liste des figures

Figure 1.1: logo PEAQOCK.....	9
Figure 1.2 : l'organigramme de l'entreprise PEAQOCK.....	10
Figure 1.3: logo OneGate.....	12
Figure 1.4: logo fraud & risk management solution.....	13
Figure 1.5: logo AML.....	14
Figure 1.6: logo PFM.....	14
Figure 1.7: logo tableau de bord dynamique.....	15
Figure 1.8: logo RegTech.....	16
Figure 1.9: logo funds.....	16
Figure 1.10: logo FIS.....	16
Figure 1.11: logo ATOS.....	17
Figure 1.12: logo HPS.....	17
Figure 1.13: liste des clients de PEAQOCK.....	18
Figure 1.14: présentation d'équipe.....	20
Figure 2.1: schéma qui démontre comment fonctionne les LLMs.....	25
Figure 2.2: Text-to-code LLMs.....	28
Figure 2.3: Text-to-Sql LLMs.....	29
Figure 2.4 Histoire des LLMs.....	31
Figure 2.5 : Classement des LLM Open.....	34
Figure 2.6: LMSYS Chatbot Arena Leaderboard.....	35
Figure 2.7: Classement des LLM de Génération de code sur le benchmark HumanEval.....	38
Figure 2.8: Classement des LLM de Génération de code sur le benchmark MBPP.....	38
Figure 2.9: Un exemple de texte vers SQL basé sur LLM sélectionné dans l'ensemble de données Spider.....	40
Figure 2.10: le processus évolutif de la recherche Text-to-SQL.....	40
Figure 2.11: Text-To-SQL on BIRD (BIg Bench for LaRge-scale Database Grounded Text-to-SQL Evaluation).....	41
Figure 2.12: Text-To-SQL benchmark on spider.....	41
Figure 2.13: classement des model LLM dans la tache Text to Sql.....	42
Figure 3.1 : logo python.....	45
Figure 3.2: language SQL.....	46
Figure 3.3: huggingface logo.....	47
Figure 3.4: Google colab logo.....	48
Figure 3.5: papers with code logo.....	48
Figure 3.6 : logo Postgres.....	52
Figure 3.7: logo Mongo DB.....	52
Figure 3.8:hugging face datasets logo.....	53
Figure 3.9: logo Google Meet.....	54
Figure 3.10: logo Google Chat.....	55
Figure 3.11: logo Docker.....	55
Figure 3.12: logo Git.....	57
Figure 3.13: flux Git.....	57
Figure 3.14: logo GitHub.....	58

Figure 3.15: logo de PyCharm IDEA..... 58

Figure 4.1 parameter efficient fine tuning..... 65

Figure 4.2 : Schéma d'une base de données pour une agence de transfert d'argent..... 72

Figure 4.3: réponse du model sur la question : Quelle est la répartition des statuts des transactions..... 74

Figure 4.4: visualisation de la réponse sur la question: : Quelle est la répartition des statuts des transactions..... 74

Figure 4.5: réponse sur la question : Pouvez-vous fournir le nombre de transactions pour chaque agent en indiquant son nom et le nombre des transactions? 75

Liste des tableaux

<i>Table 1: fiche d'identité de PEAQOCK</i>	9
<i>Table 2: distribution des données dans le dataset</i>	63
<i>Table 3: Comparaison des LLM fine-tuned</i>	71

Références

Site Web Peacock	<ul style="list-style-type: none"> • https://peacock.com/
Site web OneGate	<ul style="list-style-type: none"> • https://Onegate.ai/
Paper with code text to sql	<ul style="list-style-type: none"> • https://paperswithcode.com/task/text-to-sql
Paper with code text to code	<ul style="list-style-type: none"> • https://paperswithcode.com/task/text-to-code-generation
Dataset pour entraînement	<ul style="list-style-type: none"> • https://huggingface.co/datasets/OneGate/OGText2SQL
Organisation de OneGate sur HF	<ul style="list-style-type: none"> • https://huggingface.co/OneGate
Model Llama3-8B fine-tuned	<ul style="list-style-type: none"> • https://huggingface.co/OneGate/Llama3-OGSQL-FT-8B
Model code llama 7b fine-tuned	<ul style="list-style-type: none"> • https://huggingface.co/OneGate/OG-SQL-7B
Mistral 7B instruct fine-tuned	<ul style="list-style-type: none"> • https://huggingface.co/OneGate/OGSQL-Mistral-7B
Des articles et des ressources sur TEXT-to SQL	<ul style="list-style-type: none"> • https://medium.com/the-story-within/state-of-text-to-sql-dc3e3e4f8c64 • https://www.text2sql.ai/ • https://medium.com/pinterest-engineering/how-we-built-text-to-sql-at-pinterest-30bad30dabff • https://www.aimodels.fyi/papers/arxiv/next-generation-database-interfaces-survey-llm-based • https://arxiv.org/html/2406.08426v1
Text-To-SQL on BIRD dataset benchmarks	<ul style="list-style-type: none"> • https://paperswithcode.com/sota/text-to-sql-on-bird-big-bench-for-large-scale
Des datasets	<ul style="list-style-type: none"> • https://bird-bench.github.io/ • https://huggingface.co/datasets/b-mc2/sql-create-context • https://huggingface.co/datasets/xlangai/spider
Ressources visuelles :	<ul style="list-style-type: none"> • https://www.youtube.com/@airoundtable • https://www.youtube.com/@matthew_berman • https://www.youtube.com/@samwitteveenai • https://www.youtube.com/@freecodecamp • https://www.youtube.com/watch?v=WxQbWTRNTxY • https://www.youtube.com/@AIAnytime

Summary of the Report

This report details the process of fine-tuning a large language model (LLM) to translate natural language to SQL queries. The author, who interned at the data science and artificial intelligence company PEAQOCK, first collected and cleaned a dataset of natural language queries and their corresponding SQL translations from sources including Spider, BIRD, and Hugging Face. Next, the author evaluated several LLMs, ultimately selecting LLaMA, Mistral, and Gemma based on criteria such as model size, speed, performance, open-source availability, and ease of integration. The author chose to fine-tune the models using Parameter-Efficient Fine-Tuning (PEFT) techniques like LoRA and QLoRA, which allow for efficient model adaptation with reduced computational resources. After fine-tuning, the models were evaluated based on human feedback, with Gemma demonstrating the fastest performance, albeit with moderate accuracy. Finally, the report showcases the integration of the fine-tuned model into PEAQOCK's OneGate platform, enabling users to input natural language queries and receive SQL translations as well as data visualizations.

Résumé du rapport

Ce rapport détaille le processus de réglage fin d'un modèle linguistique de grande taille (LLM) pour traduire le langage naturel en requêtes SQL. L'auteur, qui a effectué un stage au sein de l'entreprise de science des données et d'intelligence artificielle PEAQOCK, a d'abord collecté et nettoyé un ensemble de données de requêtes en langage naturel et de leurs traductions SQL correspondantes à partir de sources telles que Spider, BIRD et Hugging Face. Ensuite, l'auteur a évalué plusieurs LLM, sélectionnant finalement LLaMA, Mistral et Gemma en fonction de critères tels que la taille du modèle, la vitesse, les performances, la disponibilité open source et la facilité d'intégration. L'auteur a choisi de peaufiner les modèles à l'aide de techniques de réglage fin efficaces en termes de paramètres (PEFT) comme LoRA et QLoRA, qui permettent une adaptation efficace du modèle avec des ressources de calcul réduites. Après le réglage fin, les modèles ont été évalués sur la base de retours humains, Gemma démontrant les performances les plus rapides, bien qu'avec une précision modérée. Enfin, le rapport présente l'intégration du modèle affiné dans la plateforme OneGate de PEAQOCK, permettant aux utilisateurs de saisir des requêtes en langage naturel et de recevoir des traductions SQL ainsi que des visualisations de données.