EDIE: Vers une interaction augmentée avec les données d'entreprise : une approche par agents et modèles de langage (LLM).

Vincent Malet [1], B. Adil Soubki[1] ,C. Arnaud Glin[1]

1: Airbus Operations SAS, CS 83172, 316 Rte de Bayonne, 31027 Toulouse

AVERTISSEMENT: Les auteurs reconnaissent la contribution de Gemini™ Flash 2.0 de Google™ dans la rédaction, la structuration et le raffinement de ce manuscrit. Bien que les idées principales, la recherche et la validation finale restent celles des auteurs, l'assistance de Gemini™ a été instrumentale tout au long du processus de rédaction.

Résumé:L'essor du Big Data a permis l'IA actuelle, notamment les grands modèles de langage (LLM), qui promettent de révolutionner l'analyse des données. Cependant, le défi majeur réside dans l'intégration de ces LLM avec les « datalakes » complexes et hétérogènes (types et accès variés) hérités du Big Data. Les LLM génériques manquent souvent des connaissances spécifiques au domaine, et leur ajustement est coûteux. Cet article présente EDIE, un agent conversationnel basé sur une architecture d'agents, conçu pour faciliter l'accès et l'analyse des données dans ces environnements. Nous décrivons les stratégies d'intégration des sources de données hétérogènes, le choix du modèle de langage optimal (performance, coût, adéquation), et un cadre d'apprentissage piloté par le feedback pour assurer l'amélioration continue d'EDIE et prévenir la régression de ses performances dans les systèmes d'entreprise.

Abstract: The Big Data Era laid the foundation for the current AI revolution by providing massive datasets. Now, AI, particularly Large Language Models (LLMs), promises to transform data analysis. However, a significant challenge lies in bridging these models with complex legacy "datalakes," characterized by diverse data types, heterogeneous access methods, and numerous analytical tools. Pre-trained LLMs often lack domain-specific knowledge, and fine-tuning can be prohibitively expensive. This paper introduces EDIE (‘Elevating flight tests and acceptance Data Intelligence for Engineering’), a prototype agent-based chatbot designed to ease data access and analysis within such environments at Airbus. We detail strategies for integrating heterogeneous data sources (REST APIs, databases, PDFs) using an agent architecture. We discuss the evaluation and selection process for suitable LLMs, comparing small open-source models with large foundation models like Google™ Gemini™, considering performance and cost. Critically, we propose a feedback-driven learning framework using benchmark datasets to ensure continuous improvement and prevent performance regression. Our findings highlight the significant performance advantages of large foundation models for complex reasoning and function calling tasks, the paramount importance of well-defined data interfaces, and the invaluable role of user feedback in guiding development within intricate enterprise settings.

Keywords:Large Language Models, LLM, Agents, Chatbot, Datalake, Enterprise AI, Data Integration, RAG, Function Calling, Feedback Loop, Human-Computer Interaction, Airbus.

1. Introduction

La prolifération des données a fourni le carburant essentiel à la révolution en cours de l'Intelligence Artificielle (IA). Les modèles d'IA avancés, entraînés sur ces ensembles de données massifs, démontrent un potentiel de transformation dans toutes les industries. Les Grands Modèles de Langage (LLM) se démarquent, offrant de nouvelles capacités pour l'interaction en langage naturel, le raisonnement complexe et potentiellement l'automatisation de flux de travail sophistiqués. Cependant, la concrétisation de ce potentiel au sein des grandes organisations nécessite souvent de combler le fossé entre ces LLM et les paysages de données complexes, souvent fragmentés, hérités de l'ère du Big Data.

Au sein d'une grande entreprise comme Airbus, une richesse d'informations réside dans de nombreux systèmes spécialisés. Les exemples clés comprennent :

- **Time Series Analytics Studio (TSAS)** : Un système critique offrant un accès à de vastes quantités de données de séries temporelles de capteurs d'aéronefs et aux métadonnées associées (configurations d'aéronefs, détails de vol, définitions de paramètres …) principalement via des API REST.
- **Enterprise Content Management (ECM)** : Le référentiel central pour les références documentaires, contenant des spécifications, des rapports, des procédures et d'autres informations cruciales non structurées ou semi-structurées.

La disponibilité de données aussi riches et interconnectées présente une opportunité significative. Les LLM, avec leurs capacités de compréhension et de génération, pourraient théoriquement permettre aux utilisateurs d'interroger, d'analyser et de synthétiser des informations couvrant ces sources disparates beaucoup plus efficacement que ne le permettent les méthodes actuelles.

Malgré cette opportunité, des obstacles importants existent :

- **Hétérogénéité des Systèmes** : Les données sont stockées et accessibles par divers moyens : API REST avec des protocoles spécifiques, bases de données SQL nécessitant des requêtes structurées, documents non structurés (PDF, rapports) et potentiellement des frameworks de calcul distribués comme Spark. L'intégration de ceux-ci nécessite diverses approches techniques.
- **Combinatoire des cas d'usage** : Les façons potentielles dont les utilisateurs pourraient vouloir combiner des données de différentes sources sont vastes.
- **Complexité de l'intégration LLM** : Les LLM et les frameworks d'agents associés représentent une pile technologique en évolution rapide. Les meilleures pratiques pour le prompting, l'intégration d'outils (appel de fonctions), la gestion du contexte et la sélection de modèles sont encore émergentes. Les LLM "modèles de fondations" manquent souvent de connaissances approfondies spécifiques au domaine (par exemple, la terminologie de l'ingénierie aéronautique, les spécificités des systèmes internes) nécessaires pour naviguer efficacement dans les données d'entreprise.

Face à ces opportunités et défis, nous avons lancé le projet EDIE en tant que prototype interne avec trois objectifs principaux :

- **Gain de Temps pour l'utilisateur** : Fournir un avantage tangible aux utilisateurs finaux en simplifiant le processus d'accès et de connexion des données à partir de plusieurs sources, réduisant la « pénibilité » souvent associée à la navigation dans l'environnement de données complexe d'Airbus.
- **Adoption et Apprentissage Technologiques** : Acquérir une expérience pratique avec les LLM, les architectures d'agents et les technologies connexes (RAG, bases de données vectorielles). Comprendre leurs capacités, leurs limites et leurs exigences d'intégration dans notre contexte spécifique. De plus, il est essentiel d'éduquer notre population d'ingénieurs sur l'application judicieuse de ces technologies dans leur travail quotidien. Cela inclut la culture d'un état d'esprit critique, l'accent sur l'importance de la vérification et le renforcement de la compréhension que ces outils sont des assistants, et non des remplacements.
- **Exploration de Nouveaux Cas d'Utilisation** : Découvrir de nouvelles façons de tirer parti des sources de données combinées. En permettant une interaction plus facile au-delà des limites du système, nous visions à découvrir des possibilités analytiques qui étaient auparavant irréalisables ou négligées.

2. Approche

Notre approche s'est concentrée sur le développement itératif, en commençant par un périmètre contraint et en élargissant progressivement les capacités en fonction de la faisabilité technique et du feedback utilisateur. Nous avons utilisé une architecture basée sur des agents où EDIE orchestre les interactions entre l'utilisateur, le LLM et divers outils conçus pour interfacer avec les systèmes backend.

2.1 Sélection d'un cas d'usage

Compte tenu des vastes possibilités combinatoires des sources de données et des besoins des utilisateurs, une sélection rigoureuse des cas d'usage initiaux était primordiale. Nous avons adopté une stratégie visant à équilibrer la complexité de la mise en œuvre avec la valeur ajoutée potentielle pour les utilisateurs.

- Périmètre des Données : Pour gérer la complexité, nous avons initialement concentré nos efforts exclusivement sur les données liées à un programme avion et un numéro de série spécifique : Cela a contraint le volume et la variété des données qu' EDIE devait gérer initialement.
- Sélection des scénarios : Nous avons priorisé les scénarios impliquant l'interrogation et la combinaison d'informations provenant d'au moins deux sources différentes, représentant des tâches courantes mais souvent chronophages pour les ingénieurs.
- Combinaison des Métadonnées de Vol et de la Documentation : « Quelle est la demande d'essai en vol associée au vol XXX du MSN YYYY ? » (Nécessite d'interroger les métadonnées TSAS et de rechercher des documents ECM).

2.2 Evaluation des modèles

Au début du projet, l'accès aux API des modèles de fondation commerciaux n'était pas encore disponible dans notre environnement. Cette contrainte a orienté notre exploration initiale vers des modèles open-source qui pourraient être déployés de manière réaliste sur le matériel disponible – ciblant des modèles exécutables sur un seul GPU NVIDIA A100 pour éviter la complexité supplémentaire des configurations d'inférence distribuée et prioriser le délai de mise sur le marché du prototype.

Nous avons évalué plusieurs modèles candidats en fonction de leurs capacités et de leur taille rapportées. Pour évaluer leur adéquation aux tâches de raisonnement et de suivi d'instructions requises par EDIE, nous avons utilisé une version réduite du benchmark MMLU[1] (Massive Multitask Language Understanding), en nous concentrant spécifiquement sur les suites pertinentes pour l'ingénierie, la logique et le raisonnement.

Les modèles testés et leurs scores sur notre benchmark MMLU élagué étaient :

Model	Quantization	Pruned MMLU Score
mistralai/Mixtral-8x7B-Instruct-v0.1	q8	0.4867
mistralai/Mistral-7B-Instruct-v0.2		0.4356
mistralai/Mistral-7B-Instruct-v0.3		0.4404
meta-llama/Llama-3-70b-Instruct	q8	0.3293

Figure 1 : Résultats MMLU élagués sur les modèles open source.

Sur la base de ces résultats initiaux et de sa taille de paramètre correspondant à nos contraintes matérielles (via la quantification 8 bits), mistralai/Mixtral-8x7B-Instruct-v0.1 a été sélectionné comme modèle initial pour le développement d'EDIE.

2.3 Interfaces des données

Un composant essentiel d'EDIE est l'ensemble des « outils » que l'agent peut utiliser pour interagir avec les systèmes de données d'entreprise. Nous avons développé des interfaces pour les principales sources de données identifiées dans nos histoires utilisateur initiales.

REST APIs (TSAS)

Le Time Series Analytics Studio (TSAS) expose principalement ses données via des API REST. Ces APIs donnent accès aux **métadonnées** (Informations sur les aéronefs, les vols, les campagnes d'essais en vol…) et **données de capteurs** : Lectures de séries temporelles pour des paramètres spécifiés pendant les vols.

Conclusions clés:

Spécificité plutôt que Généralité : Nous avons constaté que la création de plusieurs fonctions (outils) petites et spécifiques d'appel d'API pour l'agent LLM était plus fiable que la fourniture d'une seule grande fonction avec un langage de requête complexe (comme SPARQL ou un appel REST hautement paramétré). Des fonctions étroitement définies avec des objectifs clairs étaient plus faciles à sélectionner et à utiliser correctement pour le LLM.

Interface Fonctionnelle Propre : La définition de signatures de fonctions claires et explicites utilisant des indications de type (type hints), des énumérations (enum) et des ensembles de valeurs restreints (Littéral en typage Python) a considérablement amélioré la fiabilité de l'appel de fonctions. Des définitions précises du format de sortie ont également aidé le LLM à interpréter les résultats.

Bases de données (DB)

Certaines informations pertinentes, souvent organisées par les utilisateurs ou des équipes spécifiques, résidaient dans des bases de données SQL traditionnelles. Cela comprenait des tables contenant des événements spécifiques marqués avec des horodatages et des métadonnées associées.

Conclusions clés:

Minimisation des colonnes : Lors de l'interrogation de bases de données, le passage des seules colonnes nécessaires au contexte LLM était crucial. L'inclusion de colonnes superflues augmentait la probabilité que le modèle devienne confus ou interprète mal le schéma de données.

Approche par l'exemple (Few-Shot Prompting) : Fournir des exemples dans l'invite (prompt) pour la génération SQL ou l'interprétation des données était souvent précieux. Cependant, les exemples doivent être suffisamment représentatifs des attentes de l'utilisateur pour guider efficacement l'outil, sans être trop spécifiques pour limiter sa capacité à gérer des requêtes légèrement variées mais valides.

Documents PDF (ECM)

Les documents de référence des essais en vol, stockés dans l'ECM, étaient une source clé de données non structurées. Les exemples incluent :

- **Ordres de vol / Demandes d'essais en Vol** (FTR) : Documents détaillant les objectifs et les exigences pour des vols d'essai spécifiques, émis par le bureau d'études.
- **Rapports des navigants** (FCRs) : Résumés post-vol rédigés par les équipages d'essai.

Nous avons mis en œuvre un RAG (Retrieval augmented generation) pour interroger ces documents. Le périmètre était initialement limité aux documents pertinents pour un avion et une campagne d'essais en vol spécifiques.

Infrastructure : Une base de données vectorielle dédiée a été créée et peuplée grâce à un modèle d'embedding dédié.

Contraintes : Une considération importante était que certains documents sont soumis à des réglementations de contrôle des exportations, nécessitant une gestion prudente de l'accès et du traitement des données lors du traitement avec des modèles de fondation externes (c'est-à-dire dans le cloud).

Initialement, les documents manuscrits comme les Journaux de Vol, malgré leurs précieuses informations, ont été exclus du périmètre, car la reconnaissance de caractères (OCR/NLP) restait perfectibles. Cependant, les avancées récentes dans des modèles comme Gemini ont démontré la capacité à lire et à extraire des informations à partir de documents manuscrits. Cela suggère un potentiel pour élargir le périmètre documentaire d'EDIE afin d'inclure ces ressources auparavant inaccessibles.

Conclusions clés:

Compréhension sémantique : Les modèles d'embedding open-source n'ont pas toujours saisi les nuances spécifiques de la terminologie aéronautique, impactant potentiellement la pertinence de la récupération.

Stratégie de Segmentation (Chunking) : La stratégie optimale pour diviser les documents en segments pour l'embedding (taille des segments, chevauchement, méthode de division) variait en fonction du type de document (par exemple, rapports structurés par rapport à des documents procéduraux).

Limitations de l'Extraction de Données : Le parsing PDF standard et les techniques RAG ont eu du mal avec des éléments complexes comme les tableaux intégrés, les figures avec légendes, et surtout les annotations manuscrites courantes dans certains rapports, entraînant

une perte d'informations. Cela était également dû au fait que ces segments de documents étaient générés à partir de documents imprimés, ce qui introduit un degré de perte d'information avant le calcul de l'embedding.

2.3.4 Flexibilité et Contrôle

Dans les systèmes agentiques, le problème de la « Flexibilité vs Contrôle » met en évidence le compromis entre permettre à l'agent de s'adapter de manière autonome et de gérer diverses situations (flexibilité) et garantir que ses actions sont sûres, prévisibles et alignées sur les résultats souhaités (contrôle). Trouver le bon équilibre est crucial pour créer des agents puissants mais fiables.

	Moins de contrôle / Plus flexible	**Plus de contrôle / moins flexible**
REST Apis	Grandes Fonctions avec Beaucoup de Paramètres + Haute Flexibilité + Peu d'appels de fonction - Hallucinations	Petites Fonctions Spécifiques : + Sécurité et Contrôle Améliorés + Précision LLM Améliorée + Modularité / plus évolutive - Moins Flexible - Nécessite un orchestrateur
Base de données (SQL) - stratégie par l'example	Beaucoup d'Exemples Spécifiques : + Haute Précision pour des cas très spécifiques - Surapprentissage (Overfitting) aux Exemples Spécifiques	Moins d'Exemples, Plus Génériques : + bonne généralisation - moins approprié pour des cas très spécifiques

Pour le développement d'EDIE en tant que chatbot agentique, le compromis « Flexibilité vs Contrôle » se traduit directement par la manière dont nous concevons ses interactions avec les outils externes et sa capacité à générer des requêtes SQL.

Concernant les API REST : Si EDIE s'appuie sur de grandes fonctions polyvalentes, cela pourrait sembler initialement flexible, gérant potentiellement diverses requêtes utilisateur avec moins d'outils définis. Cependant, cela pourrait amener EDIE à halluciner des paramètres de fonction ou à mal interpréter les résultats. Au lieu de cela, l'adoption de fonctions plus petites et plus ciblées pour des tâches spécifiques nous a donné un plus grand contrôle sur les actions d'EDIE, a amélioré la précision des appels d'outils et a rendu le débogage plus facile, même si cela nécessite un mécanisme d'orchestration plus sophistiqué.

Concernant la Génération SQL : Si EDIE doit interroger des bases de données en utilisant une approche par l'exemple (few-shot), fournir de nombreux exemples très spécifiques pourrait le faire bien fonctionner sur ces requêtes exactes. Cependant, cela pourrait amener EDIE à sur-apprendre et à avoir du mal avec de nouveaux schémas de base de données ou des questions d'utilisateurs légèrement différentes. Par conséquent, se concentrer sur moins d'exemples, plus génériques, qui enseignent à EDIE les principes fondamentaux de la traduction du langage naturel en SQL, conduirait probablement à une capacité de génération SQL plus robuste et adaptable pour un plus large éventail de requêtes utilisateur et de structures de base de données.

2.4 Feedbacks utilisateurs et Benchmarking

Dans le développement logiciel traditionnel, les Tests d'Acceptation Utilisateur (UAT) valident les fonctionnalités. Dans les systèmes d'IA complexes comme EDIE, en particulier ceux basés sur des agents, la validation nécessite une approche différente : le benchmarking.

Nous définissons un benchmark comme une liste organisée de questions utilisateur représentatives associées à leurs réponses « de reference » ou aux résultats attendus (par exemple, un tracé spécifique, une information correcte, une séquence valide d'appels d'outils).

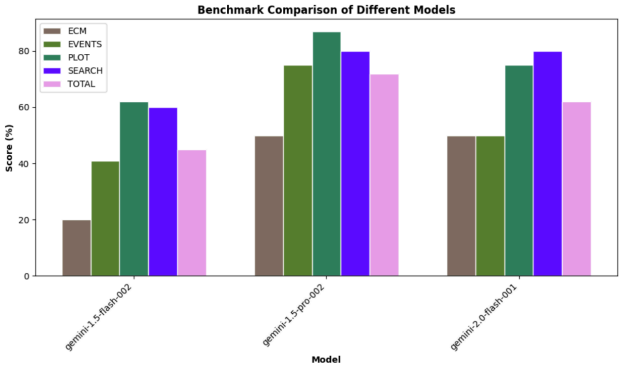

Figure 2: Benchmark de comparaison des modèles Gemini

Nous avons construit des benchmarks spécifiques pour EDIE avec deux objectifs principaux :

- *Prévention de la Régression : Au fur et à mesure que nous itérions sur la conception de l'agent, modifiions les invites, mettions à jour les outils ou expérimentions différents LLM, l'exécution du benchmark nous permettait d'identifier rapidement si les changements avaient un impact négatif sur les performances des cas d'utilisation établis.*
- *Évaluation de l'Amélioration du Système : Les benchmarks fournissaient un moyen quantitatif de mesurer l'impact des nouvelles stratégies, des modèles (par exemple, différentes techniques RAG, des méthodes de raisonnement alternatives comme ReAct) ou des modèles. Nous pouvions objectivement comparer les résultats avant et après la mise en œuvre d'un changement.*

Le feedback continu des utilisateurs pilotes, intégré parallèlement aux scores de benchmark, a fourni des informations qualitatives sur l'utilisabilité, les zones de confusion et les améliorations souhaitées.

2.5 De la faisabilité à la Fonctionnalité: Surmonter les limitation de Mixtral™ avec Gemini™ Pro

Notre version initiale d'EDIE, construite à l'aide du modèle quantifié Mixtral-8x7B, a démontré la faisabilité mais a également mis en évidence des limitations significatives, en particulier lorsque les requêtes utilisateur devenaient plus complexes :

- Hallucinations d'Appel de Fonction : Le modèle a parfois tenté d'appeler des fonctions avec des paramètres incorrects ou a halluciné des valeurs de paramètres non dérivées de l'historique de la conversation ou des données récupérées.
- Erreurs d'Interprétation des Résultats : Après avoir réussi à appeler un outil/API, le modèle a occasionnellement mal interprété les résultats retournés, conduisant à des réponses finales incorrectes.
- Limitations de Raisonnement : Le modèle a montré des faiblesses dans le raisonnement complexe en plusieurs étapes requis pour l'exécution d'agent de type Chain-of-Thought ou ReAct, échouant souvent à décomposer les problèmes efficacement ou à maintenir le contexte à travers les étapes.
- Sortie Non Déterministe : Même avec la même invite et le même contexte, le modèle produisait occasionnellement des sorties variables, ce qui rendait difficile d'assurer un comportement cohérent et fiable pour les tâches critiques.

Par la suite, nous avons eu accès à la famille de modèles de fondation Gemini™ de Google™ via API. La migration du composant LLM central d'EDIE vers Gemini™ Pro a entraîné une amélioration spectaculaire des performances :

- ➔ Appel de Fonction Natif : Les capacités natives robustes d'appel de fonction/outil de Gemini™ ont rendu l'intégration avec nos outils d'API significativement plus fiable et presque sans faille par rapport aux techniques de prompting structurées requises pour Mixtral™.
- ➔ Raisonnement Amélioré : Gemini™ a démontré des capacités de raisonnement largement supérieures, gérant avec succès des requêtes multi-étapes plus complexes et présentant un comportement de type ReAct plus robuste.

Cette transition a souligné un écart de capacité significatif entre le modèle open-source moyen déployé (bien qu'un modèle puissant comme Mixtral™) et un grand modèle de fondation de pointe pour les tâches nécessitant une utilisation fiable des outils et un raisonnement complexe au sein d'un cadre agentique.

7. Croisement des données du datalake ; découvertes et opportunités

L'intégration et la corrélation d'informations provenant de diverses sources au sein d'un datalake d'entreprise complexe ont toujours été un défi majeur. Les anciennes approches d'intégration de données reposent souvent sur un effort manuel pour concevoir et mettre en œuvre des pipelines statiques prédéfinis (par exemple, processus ETL). Ces méthodes nécessitent généralement des connaissances informatiques considérables, une expertise dans les langages de requête spécifiques ou les frameworks (comme SQL ou SPARK), et sont souvent mal adaptées à l'exploration dynamique et ad hoc à travers des systèmes hétérogènes. Elles impliquent généralement un investissement initial important pour définir les croisements de données potentiels, limitant la flexibilité lorsque de nouvelles questions se posent.

Les systèmes agentiques comme EDIE offrent un changement de paradigme pour le croisement des données au sein du datalake. Tirant parti des capacités de raisonnement des LLM et d'un ensemble d'outils flexible, EDIE peut interpréter dynamiquement les requêtes utilisateur exprimées en langage naturel. Il peut orchestrer de manière autonome les interactions avec plusieurs sources de données – API REST, bases de données, référentiels de documents – découvrant et récupérant des informations pertinentes à la volée en fonction de la requête de l'utilisateur.

Notamment, l'intégration d'un REPL Python et de capacités de traçage dynamique offre un moyen encore plus flexible d'accéder, de manipuler et de présenter des données, améliorant la capacité de l'utilisateur à explorer et à visualiser les résultats.

Crucialement, cette approche basée sur des agents peut assouplir la contrainte traditionnelle de devoir centraliser toutes les données pertinentes dans un seul lac de données ou datalake. En interagissant avec les sources de données directement via leurs interfaces natives (API, EndPoint), l'agent permet aux données de rester là où elles résident. Cela réduit considérablement le besoin de copie de données étendue et de processus ETL, en particulier pour les cas d'utilisation d'analyse interactionnelle, et s'adapte plus facilement aux architectures de données d'entreprise intrinsèquement distribuées ou fédérées où les données restent éparses entre les systèmes.

Cela permet aux utilisateurs sans expertise technique approfondie de poser des questions complexes qui couvrent différents systèmes (par exemple, « Trouvez le document FTR lié au vol où le paramètre X a dépassé le seuil Y »), obtenant des réponses synthétisées sans avoir besoin de comprendre les méthodes d'accès sous-jacentes ou les schémas de données.

De plus, une force clé des systèmes agentiques réside dans leur capacité émergente à répondre à des cas d'utilisation inattendus ou nouveaux en combinant dynamiquement

leurs outils disponibles de manière non explicitement pré-programmée. À mesure que le nombre d'outils discrets (interfaçant avec différentes sources de données ou fonctionnalités) augmente de manière linéaire, le nombre potentiel de façons dont ces outils peuvent être chaînés et combinés pour répondre à des requêtes complexes en plusieurs étapes augmente de manière presque exponentielle. Cette puissance combinatoire élargit considérablement la portée des questions que le système peut potentiellement gérer. Cependant, cette flexibilité inhérente présente également un défi important : l'orchestration efficace de ces combinaisons d'outils et le maintien d'un contrôle robuste sur le processus de raisonnement de l'agent deviennent critiques. S'assurer que l'agent sélectionne la séquence appropriée d'outils, transmet correctement les informations entre les étapes et évite les chemins absurdes ou erronés (« se perdre » dans l'espace combinatoire) est primordial pour des résultats fiables et dignes de confiance.

La capacité de l'agent à comprendre le contexte, à enchaîner les appels d'outils et à intégrer les résultats fournit une interface intelligente, dynamique et intuitive pour l'exploration et la découverte de données à travers des ensembles de données auparavant cloisonnés. Cependant, il est important de noter que les approches traditionnelles restent essentielles pour les tâches de traitement, de nettoyage et de transformation de données à grande échelle et orientées par lots (traitements ETL) où des flux de travail structurés et reproductibles opérant sur des ensembles de données massifs sont requis, et où les relations de données sont bien définies et stables. Les systèmes agentiques excellent dans la couche d'exploration et d'intégration ad hoc.

8. Conclusion

Le projet EDIE a démontré avec succès le potentiel de l'utilisation de chatbots basés sur des agents et alimentés par des LLM pour améliorer l'accessibilité et l'analyse des données au sein d'un environnement d'entreprise complexe caractérisé par des datalakes hérités. Grâce au prototypage et à l'itération, nous avons recueilli plusieurs conclusions et enseignements clés :

Le Feedback Utilisateur est Indispensable : Dans un domaine complexe comme l'ingénierie aéronautique chez Airbus, une boucle de feedback étroite avec les utilisateurs finaux était cruciale. Leurs informations ont guidé la priorisation des cas d'utilisation, identifié les lacunes dans la compréhension du domaine et validé l'utilité pratique du prototype.

Le Choix du Modèle Est Significatif : Alors que les modèles open-source moyens comme Mixtral™ offraient un point de départ viable sous contrainte, les grands modèles de fondation de pointe comme Google™ Gemini™ ont démontré des performances substantiellement meilleures pour les tâches centrales de l'agent de raisonnement, de suivi d'instructions, et particulièrement d'appel de fonction fiable. Pour les applications d'agent d'entreprise complexes, les capacités de ces modèles offrent actuellement un avantage distinct.

La Qualité de l'Interface est Primordiale : La fiabilité de l'ensemble du système dépend fortement de la qualité des interfaces (outils) connectant l'agent aux systèmes de données sous-jacents. De petites fonctions dédiées avec des signatures propres et explicites et des schémas de données bien définis se sont avérées beaucoup plus efficaces et fiables pour l'interaction LLM que les grandes interfaces génériques. Le principe « entrées poubelles, sorties poubelles » (Garbage-in, garbage-out) s'applique fortement aux données fournies au LLM via les outils.

Plus spécifiquement concernant le défi du croisement des données du datalake, EDIE offre des avantages significatifs par rapport aux méthodes traditionnelles. En agissant comme un intermédiaire intelligent, il protège les utilisateurs de la complexité de l'interaction avec divers systèmes backend (API REST, bases de données, magasins de documents). Son interface en langage naturel permet aux utilisateurs de formuler des requêtes complexes qui couvrent plusieurs sources de données sans nécessiter de compétences techniques spécialisées dans l'interrogation de chaque système individuellement. L'architecture d'agent, pilotée par les capacités de raisonnement et d'appel de fonctions du LLM, permet la récupération et la synthèse dynamiques des données en fonction du besoin immédiat de l'utilisateur, facilitant la découverte de connexions et d'insights à travers des informations auparavant cloisonnées qui étaient auparavant difficiles ou chronophages à découvrir par des pipelines statiques ou une investigation manuelle.

EDIE sert de preuve de concept précieuse, soulignant à la fois la promesse et les défis pratiques du déploiement d'agents LLM contre des données d'entreprise hétérogènes. Les travaux futurs se concentreront sur l'élargissement de la gamme d'outils intégrés, l'affinage du processus RAG pour une meilleure compréhension sémantique, l'amélioration des mécanismes de feedback pour l'apprentissage continu et l'abordage des exigences d'entreprise comme la sécurité et la gouvernance.

8. References

[1] https://paperswithcode.com/dataset/mmlu

10. Glossary

IA: *Intelligence Artificielle*

ETL: *Extract Transform Load*

LLM: *Large Language Model*

MCP: *Model Context Protocol*

MLLU: *Massive Multitask Language Understanding*

REST: *Representational State Transfer*

REPL: *Read, Evaluate, Print, and Loop. Used to communicate with the Python Interpreter for generating and evaluating code dynamically.*