Agent conversationnel pour l'interrogation de la base de données Open Food Facts

Description et planification du projet

Travail présenté à Luc Lamontagne
IFT-6005 - Projet intégrateur

réalisé par

Alain Boisvert

994 029 313

30 janvier 2025 9h30

Table des matières

1 Description du problème

L'accès aux bases de données relationnelles repose traditionnellement sur des langages de requête structurés (par exemple, SQL), excluant ainsi les utilisateurs dépourvus de compétences techniques. Cette contrainte limite l'exploitation optimale des données, particulièrement dans les domaines nécessitant une accessibilité grand public.

Un cas d'usage illustratif est la consultation d'informations nutritionnelles. Des plateformes comme *Open Food Facts*[1] agrègent des millions de fiches produits détaillées (composition, apports nutritifs, certifications), mais leur interface requiert des requêtes manuelles ou des filtres prédéfinis. Pour un consommateur souhaitant poser des questions plus complexes (*« Quels en-cas sans allergènes ont un Nutri-score A ? »*), l'expérience utilisateur reste laborieuse.

Ce projet explore l'utilisation des grands modèles de langage (LLM) pour faciliter l'accès aux bases de données. Les LLM peuvent comprendre les questions posées en langage naturel et les convertir en requêtes de base de données. Cette approche présente deux avantages majeurs :

— **Simplification de l'accès aux données :** Les utilisateurs peuvent poser des questions dans leur langage habituel, sans connaissances techniques.
— **Amélioration de la qualité des réponses :** En accédant directement aux sources de données structurées, le système peut fournir des réponses plus précises et fiables.

L'objectif de ce projet est de développer un agent conversationnel capable de répondre à des questions en langage naturel en interrogeant la base de données de produits alimentaires d'Open Food Facts.

2 Revue de littérature

L'interaction en langage naturel avec les bases de données représente un défi majeur en intelligence artificielle. Les récents progrès des LLM ouvrent de nouvelles perspectives pour permettre aux utilisateurs de formuler des requêtes complexes sans connaître le langage SQL. Cette « très brève » revue de littérature examine les avancées récentes dans ce domaine, en se concentrant particulièrement sur les systèmes de conversion texte-vers-SQL utilisant des LLM.

— L'étude de Hong et al. [3] présente une revue complète de la conversion texte-vers-SQL basée sur les LLM et des défis techniques. Les auteurs présentent une analyse des avancées récentes dans la conversion texte-vers-SQL basée sur les LLM. Les auteurs fournissent également une introduction détaillée aux jeux de données et aux métriques conçus pour évaluer les systèmes de conversion texte-vers-SQL.
— Mohammadjafari et al. [6] examinent l'évolution des systèmes de conversion texte-vers-SQL, depuis les premiers modèles basés sur des règles jusqu'aux approches avancées utilisant les LLM, les jeux de test, les méthodes d'évaluation et les métriques d'évaluation. Ils discutent des techniques de fine-tuning, d'apprentissage zero-shot et few-shot, et de l'augmentation des données. Ils étudient aussi le rôle de l'intégration des graphes de connaissances pour améliorer la précision contextuelle et la liaison des schémas dans ces systèmes.
— Zhu et al. [7] classifient les approches de conversion texte-vers-SQL basées sur les LLM en groupes selon leurs stratégies d'entraînement : ingénierie des prompts, fine-tuning, pré-entraînement et agents. Ils présentent également un résumé complet des jeux de données et des métriques d'évaluation.
— Gao et al. [2] proposent DAIL-SQL, une méthode d'ingénierie des prompts pour améliorer la conversion de textes vers SQL basée sur les LLM. Les auteurs étudient l'ingénierie des prompts, incluant la représentation des questions, la sélection et l'organisation des exemples. Leur méthode atteint une précision remarquable de 86,6% sur le jeu de test

1. https://world.openfoodfacts.org/

Spider, démontrant qu'une ingénierie des prompts peut significativement améliorer la qualité des requêtes SQL générées. Ils explorent également le potentiel des LLM open source dans la conversion texte-vers-SQL et le fine-tuning supervisé pour améliorer les performances.

— Biswal et al. [1] présentent TAG (*Table-Augmented Generation*, génération augmentée par table), une nouvelle approche pour interroger les bases de données en langage naturel. Ce modèle en trois étapes (synthèse de requêtes, exécution et génération de réponses) combine les capacités de raisonnement sémantique des LLM avec la puissance des systèmes de bases de données. Cette approche permet de traiter des questions plus complexes nécessitant une combinaison de raisonnement textuel, de calculs et de connaissances générales. TAG surmonte les limitations des approches existantes comme Text2SQL et RAG (*Retrieval-Augmented Generation*), offrant une amélioration de performance de 20-65% dans les tests.

— Li et al. [4] introduisent le jeu de test BIRD, conçu pour répondre aux limitations des jeux de test de conversion texte-vers-SQL existants (comme Spider et WikiSQL). BIRD contient 12 751 paires texte-vers-SQL et 95 bases de données d'une taille totale de 33,4 Go, couvrant 37 domaines professionnels. Les auteurs se concentrent sur les défis liés aux données bruitées, aux connaissances externes et à l'efficacité SQL. Les résultats montrent que même des LLM avancés comme GPT-4 n'atteignent que 54,89% de précision d'exécution sur BIRD, significativement inférieur à la performance humaine (92,96%).

— Malekpour et al. [5], de Polytechnique Montréal, proposent une approche de routage LLM pour la conversion texte-vers-SQL, qui sélectionne dynamiquement le LLM le plus rentable capable de générer du SQL précis pour chaque requête. Ils présentent deux stratégies de routage (basées sur le score et sur la classification) qui atteignent une précision comparable au LLM le plus performant tout en réduisant les coûts, facilitant l'entraînement et permettant une inférence efficace.

Ces articles démontrent l'évolution rapide de la conversion texte-vers-SQL utilisant les LLM. Ils mettent en évidence l'importance des jeux de test, d'une ingénierie des prompts, et des approches innovantes pour rendre des systèmes plus performants.

Ces travaux sont pertinents pour le projet d'agent conversationnel pour Open Food Facts. L'approche de DAIL-SQL pourrait être adaptée pour optimiser la conversion des questions nutritionnelles en requêtes SQL précises, tandis que l'approche TAG pourrait améliorer la qualité des réponses en combinant les données structurées avec le raisonnement sémantique. Les stratégies de routage LLM proposées par l'équipe de Polytechnique Montréal offrent des pistes pour optimiser les coûts. De plus, les méthodologies d'évaluation présentées dans ces articles, notamment dans le contexte de BIRD, pourront guider la création d'un jeu de test spécifique aux requêtes sur les produits alimentaires.

Ces articles seront étudiés plus en détail au cours des prochaines semaines dans le cadre de ce projet afin d'identifier les meilleures pratiques à adopter.

3 Approche proposée

Le système proposé repose sur une architecture modulaire pour faciliter le développement, l'entraînement et l'évaluation, qui comprendra quatre composants clés interconnectés :

— **Module de dialogue :** Responsable de maintenir le contexte conversationnel et de gérer les interactions avec l'utilisateur.

— **Conversion texte-SQL :** Transforme les questions en langage naturel en requêtes SQL.

— **Connecteur de base de données :** Interface avec une base de données pour l'exécution des requêtes.

— **Générateur de réponses :** Transforme les résultats bruts en réponses naturelles et contextuelles.

Un LLM pré-entraîné pour le dialogue et capable d'utiliser des outils sera utilisé. Ce modèle sera probablement Qwen2-7B-Instruct [2] pour ses capacités natives d'appel de fonctions.

La conversion texte-SQL sera réalisée en plusieurs étapes, inspirées de Biswal et al. [1]. L'utilisation du one-shot et few-shot learning pour améliorer la performance et un outil comme sqlglot [3] pour valider la syntaxe des requêtes SQL sera considérée.

Pour la base de données, une version simplifiée de la base de données Open Food Facts, stockée au format Parquet, sera utilisée. Ces données seront converties en une base DuckDB pour faciliter les requêtes SQL.

Pour les validations, un ensemble de questions de référence, inspiré de BIRD, sera créé pour les données d'Open Food Facts.

Un développement incrémental est prévu, en commençant par une approche simple et évolutive vers un système plus complet. Voici les étapes proposées :

— Créer un système conversationnel avec un LLM open-source
— Convertir les questions en requêtes SQL
— Créer un outil pour interroger la base de données via une requête SQL
— Générer une réponse à partir des résultats de la requête
— Évaluer la qualité des réponses et le temps d'exécution
— Optimiser le système pour améliorer les performances (par exemple, optimiser les prompts, faire du one-shot ou few-shot learning)

Le système sera développé en Python, en utilisant des bibliothèques comme Hugging Face Transformers, Ollama, et DuckDB sur un MacBook Pro M1 16 Go.

Cette approche modulaire permettra d'itérer rapidement tout en maintenant un haut niveau de fiabilité.

4 Données utilisées

Les données Open Food Facts au format Parquet est disponible sur 🤗 Hugging Face [4]. Ce format orienté colonnes est particulièrement efficace pour le stockage et l'analyse de grands jeux de données sur un ordinateur personnel avec une RAM limitée.

La base complète contient 3 601 655 produits alimentaires décrits par 109 colonnes, incluant :

— Informations générales : nom, marque, quantité
— Nutrition : nutriments, vitamines, minéraux
— Composition : ingrédients, allergènes, additifs
— Certifications : labels (bio, sans gluten), scores (Nutri-score, Eco-score)
— Origine et distribution : pays de fabrication, lieux de vente

Les données se présentent sous différents formats (listes, chaînes de caractères, nombres) avec une proportion importante de valeurs manquantes. Par exemple, l'Eco-score est absent dans 80% des entrées.

La figure 1 présente la distribution de la complétude des colonnes de la base de données. On observe une distribution bimodale avec un grand nombre de colonnes très complètes ($> 95\%$) et un second groupe de colonnes peu renseignées ($< 30\%$).

Pour ce projet, un sous-ensemble de 94 802 produits alimentaires canadiens sera utilisé afin de réduire les temps de traitement.

2. https://huggingface.co/Qwen/Qwen2-7B-Instruct
3. https://github.com/tobymao/sqlglot
4. https://huggingface.co/datasets/openfoodfacts/product-database

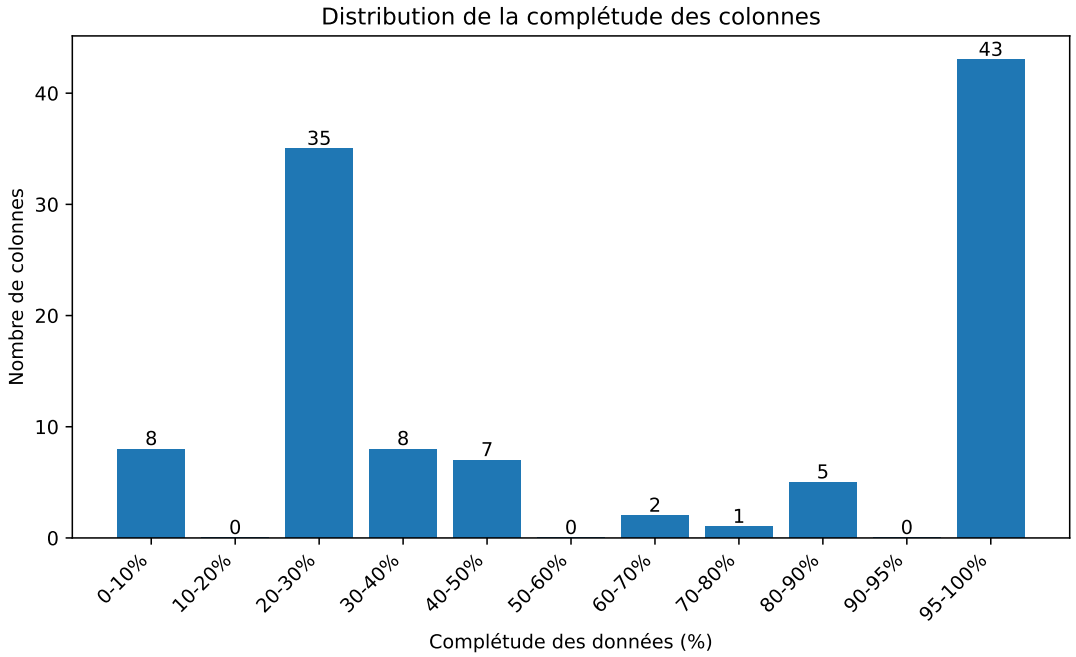

FIGURE 1 – Distribution de la complétude des colonnes de la base de données

5 Évaluation du système

L'évaluation du système s'appuiera sur trois métriques principales, inspirées des articles cités précédemment, tout en étant adaptées aux spécificités des données nutritionnelles :

1. **Précision d'exécution (EX)** : Cette métrique, adoptée dans la littérature récente [4], évalue si les requêtes SQL générées produisent les résultats attendus. Pour notre cas d'usage, un jeu de test de 100 questions nutritionnelles courantes avec leurs requêtes SQL de référence et les résultats attendus sera créé. La précision sera calculée en comparant les résultats d'exécution des requêtes générées avec ceux des requêtes de référence, plutôt qu'en comparant directement le code SQL. Cette approche permet d'accepter des formulations SQL différentes tant qu'elles produisent les mêmes résultats.
2. **Taux de couverture des données manquantes (TCM)** : Cette métrique évalue la capacité du système à fournir des réponses pertinentes malgré l'absence de certaines données. Trois stratégies sont employées :
   — Utilisation d'attributs alternatifs : par exemple, estimer le Nutri-Score à partir des valeurs nutritionnelles
   — Reformulation des requêtes pour exploiter les données disponibles
   — Communication claire des limitations et du niveau de confiance des réponses
   Le TCM sera évalué sur le même jeu de test de 100 questions, en calculant le pourcentage de cas où le système parvient à fournir une réponse exploitable malgré des données manquantes. Une réponse est considérée exploitable si elle utilise avec succès au moins une des trois stratégies mentionnées. Cette approche permet d'optimiser l'utilité des réponses même avec une base de données incomplète.
3. **Temps de réponse moyen (TRM)** : Cette métrique mesure le temps total nécessaire pour traiter une requête, depuis la réception de la question en langage naturel jusqu'à la génération de la réponse finale. Elle inclut :
   — Le temps de conversion de la question en SQL
   — Le temps d'exécution de la requête sur la base de données
   — Le temps de génération de la réponse en langage naturel
   Le TRM sera mesuré sur une configuration matérielle standardisée pour assurer la comparabilité des résultats au fil du développement. Cette métrique est particulièrement im-

portante pour évaluer l'utilisabilité du système dans un contexte réel.

Ces métriques seront évaluées sur différentes catégories de questions (simples, complexes, multiples critères) pour assurer une évaluation complète du système.

Ces tests d'évaluation seront intégrés tout au long du développement pour guider les améliorations du système.

6 Tâches à faire

Cette planification de projet (235h) suit une approche incrémentale, permettant de développer et tester chaque composant de manière itérative :

1. **Préparation et configuration (30h)**
   — Mise en place de l'environnement de développement (5h)
   — Préparation de la base de données Open Food Facts canadienne (5h)
   — Créer un jeu de test de 100 questions de référence (10h)
   — Implémentation des scripts d'évaluation des trois métriques (10h)
2. **Développement du système de base (75h)**
   — Implémentation du module de dialogue avec Qwen2-7B-Instruct (20h)
   — Développement de la conversion texte-SQL de base (25h)
   — Création du connecteur de base de données DuckDB (5h)
   — Implémentation du générateur de réponses simples (25h)
3. **Rapport de mi-session (25h)**
   — Analyse des résultats (10h)
   — Rédaction du rapport de mi-session (15h)
4. **Optimisations et fonctionnalités avancées (60h)**
   — Amélioration de la gestion des données manquantes (15h)
   — Implémentation du one-shot et few-shot learning (20h)
   — Optimisation des prompts et des requêtes SQL (15h)
   — Sélection dynamique de LLM pour accroître la performace et réduire les coûts (10h)
5. **Documentation et finalisation (25h)**
   — Rédaction du rapport final (15h)
   — Préparation de la présentation orale (5h)
   — Nettoyage du code et de la documentation du dépôt GitHub (5h)

Cette planification inclut des jalons de validation à chaque étape majeure pour s'assurer que le développement reste aligné avec les objectifs du projet. Les heures allouées tiennent compte des défis potentiels liés à l'intégration des composants et à la gestion des données incomplètes d'Open Food Facts.

Références

[1] Asim Biswal, Liana Patel, Siddarth Jha, Amog Kamsetty, Shu Liu, Joseph E Gonzalez, Carlos Guestrin, and Matei Zaharia. Text2SQL is Not Enough : Unifying AI and Databases with TAG. *arXiv preprint arXiv :2408.14717*, 2024. URL https://arxiv.org/abs/2408.14717.

[2] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. Text-to-SQL Empowered by Large Language Models : A Benchmark Evaluation. *arXiv preprint arXiv :2308.15363*, 2023. URL https://arxiv.org/abs/2308.15363.

[3] Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. Next-Generation Database Interfaces : A Survey of LLM-based Text-to-SQL. *arXiv preprint arXiv :2406.08426*, 2024. URL https://arxiv.org/abs/2406.08426.

[4] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. Can LLM Already Serve as A Database Interface? A Big Bench for Large-Scale Database Grounded Text-to-SQLs. *Advances in Neural Information Processing Systems*, 36, 2024. URL https://arxiv.org/abs/2305.03111.

[5] Mohammadhossein Malekpour, Nour Shaheen, Foutse Khomh, and Amine Mhedhbi. Towards Optimizing SQL Generation via LLM Routing. *arXiv preprint arXiv :2411.04319*, 2024. URL https://arxiv.org/abs/2411.04319.

[6] Ali Mohammadjafari, Anthony S Maida, and Raju Gottumukkala. From Natural Language to SQL : Review of LLM-based Text-to-SQL Systems. *arXiv preprint arXiv :2410.01066*, 2024. URL https://arxiv.org/abs/2410.01066.

[7] Xiaohu Zhu, Qian Li, Lizhen Cui, and Yongkang Liu. Large Language Model Enhanced Text-to-SQL Generation : A Survey. *arXiv preprint arXiv :2410.06011*, 2024. URL https://arxiv.org/abs/2410.06011.