Résumé de *Bridging the gap : Enabling natural language queries for NoSQL databases through text-to-NoSQL translation*

Jinwei Lu *et al.* (2025)

1. Contexte et motivation

Au fur et à mesure que les bases de données NoSQL gagnent en popularité pour gérer des volumes importants de données non structurées et semi-structurées, il apparaît une difficulté majeure pour les utilisateurs non techniques : l'écriture de requêtes NoSQL. Contrairement au SQL, qui dispose de standards largement enseignés, chaque système NoSQL a sa propre syntaxe et ses propres concepts (collections, documents, agrégations, etc.), ce qui crée une barrière d'entrée pour les utilisateurs sans compétences techniques. :contentReference[oaicite :0]index=0

Les auteurs identifient un besoin crucial d'interfaces conviviales qui permettent à un utilisateur de formuler des intentions en langage naturel pour obtenir des résultats à partir d'une base NoSQL — une idée inspirée des progrès des systèmes Text-to-SQL pour les bases relationnelles. :contentReference[oaicite :1]index=1

L'objectif principal de l'article est donc d'**introduire et définir la tâche Text-to-NoSQL**, qui consiste à traduire automatiquement une requête en langage naturel (NLQ) en une requête NoSQL valide et exécutable, tout en évaluant la qualité de cette traduction de manière fiable. :contentReference[oaicite :2]index=2

2. Contributions principales

Les contributions majeures de l'article sont les suivantes : :contentReference[oaicite :3]index=3

— La **formulation explicite de la tâche Text-to-NoSQL**, qui pose les fondations d'une nouvelle ligne de recherche en interaction naturelle avec les systèmes NoSQL. :contentReference[oaicite :4]index=4
— La conception d'un **grand dataset open-source appelé TEND (Text-to-NoSQL Dataset)**, construit automatiquement pour entraîner et évaluer des modèles capables de traduire des requêtes NL en requêtes NoSQL. :contentReference[oaicite :5]index=5
— L'introduction du cadre multi-étapes **SMART (Small Language Model + Retrieval-Augmented Generation)**, combinant de petites architectures de modèles de langage avec des mécanismes de récupération d'information pour améliorer la traduction. :contentReference[oaicite :6]index=6
— La définition d'une **série de métriques d'évaluation complètes**, tenant à la fois compte de la fidélité syntaxique des requêtes produites et de leurs résultats d'exécution. :contentReference[oaicite :7]index=7
— La réalisation d'**expérimentations étendues** démontrant l'efficacité du cadre proposé et établissant un benchmark pour les travaux futurs. :contentReference[oaicite :8]index=8

3. Construction du dataset TEND

Un des apports clés de l'article est la création du dataset **TEND**. Ce jeu de données contient un grand nombre de paires associant des requêtes en langage naturel à leurs équivalents NoSQL. :contentReference[oaicite :9]index=9

La construction du dataset repose sur une **procédure automatisée** qui exploite des schémas de base de données et génère des requêtes linguistiques plausibles via des modèles de langage, puis produit les requêtes NoSQL correspondantes en simulant des transformations et des règles prédéfinies. :contentReference[oaicite :10]index=10

Cette approche présente plusieurs avantages :

— Elle permet de générer un volume significatif de données d'entraînement et de test, ce qui est essentiel pour entraîner des modèles robustes.
— Elle réduit le besoin de collecte manuelle de données, souvent coûteuse et sujette à erreurs.
— Elle fournit des exemples variés couvrant différents types de structures NoSQL (documents, clés-valeurs, etc.). :contentReference[oaicite :11]index=11

4. Cadre de traduction SMART

Le framework **SMART** (Small Language Model + Retrieval-Augmented Generation) est conçu pour faire face à la complexité de la traduction en combinant des approches complémentaires : :contentReference[oaicite :12]index=12

- Une architecture de **petits modèles de langage (SLM)** capable de comprendre et transformer des motifs linguistiques simples.
- Une stratégie de **récupération augmentée par génération (RAG)** qui exploite des documents, schémas de base de données ou exemples similaires pour enrichir le contexte de traduction. :contentReference[oaicite :13]index=13

L'idée centrale est de ne pas se reposer uniquement sur une génération directe depuis un modèle, mais de **combiner l'apprentissage statistique avec des morceaux de connaissances récupérés** au moment de la requête, afin d'améliorer la précision — notamment dans les cas complexes impliquant des structures imbriquées ou des opérations agrégées. :contentReference[oaicite :14]index=14

5. Mécanismes d'évaluation rigoureux

Pour juger de façon fiable la qualité des modèles Text-to-NoSQL, les auteurs soulignent qu'il faut aller au-delà du seul score de correspondance syntaxique. Ils proposent donc des métriques qui mesurent : :contentReference[oaicite :15]index=15

- La **qualité syntaxique** des requêtes générées (similitude avec la requête cible, respect des contraintes de langage NoSQL).
- La **qualité sémantique**, c'est-à-dire si la requête produite, une fois exécutée sur une base de données, donne bien les résultats attendus correspondant à la requête en langage naturel. :contentReference[oaicite :16]index=16

Cette double approche permet de mieux refléter les défis réels de la traduction, car une requête peut être syntaxiquement correcte mais produire des résultats incorrects si elle ne reflète pas exactement l'intention exprimée. :contentReference[oaicite :17]index=17

6. Expérimentations et résultats

Les expérimentations réalisées dans l'article montrent que le cadre SMART, lorsqu'il est entraîné sur le dataset TEND, **surpasse plusieurs approches de

référence** utilisées comme baselines. :contentReference[oaicite :18]index=18

Les auteurs observent des améliorations significatives dans la capacité du système à générer des requêtes NoSQL qui sont non seulement syntaxiquement valides, mais aussi sémantiquement alignées avec les intentions des requêtes en langage naturel. :contentReference[oaicite :19]index=19

Ces résultats établissent un **benchmark initial pour la tâche Text-to-NoSQL**, fournissant des points de comparaison pour de futures méthodes ou améliorations. :contentReference[oaicite :20]index=20

7. Limites et perspectives futures

L'article reconnaît plusieurs limitations, dont :

— La dépendance à des jeux de données générés automatiquement, qui peuvent ne pas entièrement représenter les requêtes réelles d'utilisateurs finaux. :contentReference[oaicite :21]index=21

— La difficulté à gérer des requêtes extrêmement complexes ou des structures NoSQL très hétérogènes. :contentReference[oaicite :22]index=22

Pour les recherches futures, les auteurs suggèrent notamment :

— L'extension du dataset à des scénarios réels et des schémas plus variés.

— L'amélioration des composants RAG pour intégrer des connaissances plus riches (schémas, contraintes, ontologies). :contentReference[oaicite :23]index=23

8. Conclusion

Cet article pose les premières bases d'un domaine émergent, en offrant à la communauté un framework, un dataset, des métriques et des résultats expérimentaux pour encourager l'adoption de requêtes naturelles sur des bases NoSQL. Il illustre ainsi une étape importante vers des interactions plus intuitives et accessibles avec des systèmes de données complexes. :contentReference[oaicite :24]index=24

9. Utilisation de l'article dans le projet

Pour analyser les performances du modèle, on décide de travailler avec les 6 métriques mentionnées dans l'article : Metric_ExactMatch, Metric_QueryStagesMatch, Metric_QueryFieldsCoverage, Metric_ExecutionAccuracy, Metric_ExecutionFieldsMatch, Metric_ExecutionValueMatch

Résumé de

Towards User-Friendly NoSQL : A Synthetic Dataset Approach and Large Language Models for Natural Language Query Translation

Alessandro Tola (2024), Politecnico di Torino

1. Introduction et contexte

L'évolution du traitement de données à grande échelle a vu une transition progressive des bases de données relationnelles vers des systèmes NoSQL plus adaptés aux données non structurées ou semi-structurées. Bien que puissants, ces systèmes présentent une complexité syntaxique qui rend difficile l'interrogation directe par des utilisateurs non techniques. C'est dans ce contexte que se situe la thèse de master d'Alessandro Tola, qui vise à faciliter l'accès aux bases de données NoSQL via des requêtes en langage naturel. :contentReference[oaicite :1]index=1

La thèse reconnaît qu'il n'existe pas de dataset adapté à la tâche de traduction de requêtes en langage naturel vers NoSQL et que les interfaces actuelles sont peu conviviales pour des utilisateurs sans expertise technique. L'objectif est donc double : créer un dataset synthétique pour la tâche NL-to-NoSQL et exploiter des **Large Language Models (LLMs)** pour la traduction de requêtes. :contentReference[oaicite :2]index=2

2. Création d'un dataset synthétique

Une contribution centrale de cette recherche est le développement d'une méthode automatisée pour créer un dataset synthétique adapté à la traduction de requêtes en langage naturel vers des requêtes NoSQL. En l'absence d'un dataset existant pour ce cas d'usage, l'auteur s'appuie sur plusieurs techniques pour générer des paires de données pertinentes :

— **Inspiration du dataset NL-to-SQL existant**, en particulier WikiSQL, qui contient des paires question-requête SQL.
— **Utilisation de templates** de requêtes (Query templates) et de formulations en langage naturel (NL templates) pour produire automatiquement des exemples variés.
— **Stratégies d'augmentation des données** afin d'accroître la diversité

et la couverture des exemples sans recourir à l'annotation manuelle, coûteuse en temps et ressources. :contentReference[oaicite :3]index=3

Cette approche synthétique permet de générer un dataset bien structuré, diversifié et approprié pour entraîner des modèles à accomplir la tâche de traduction NL-to-NoSQL. :contentReference[oaicite :4]index=4

3. Fine-tuning de modèles de langage (LLMs)

Une fois le dataset synthétique défini, la thèse explore l'utilisation de **grands modèles de langage (Large Language Models)** pour apprendre la traduction de requêtes en langage naturel vers des requêtes NoSQL.

L'auteur met en œuvre un processus d'**affinage supervisé (Supervised Fine-Tuning)** en utilisant des techniques **PEFT (Parameter-Efficient Fine-Tuning)** via QLoRA. Ce type d'affinage permet de :

— Adapter un modèle LLaMA2 pré-entraîné de façon efficace pour la tâche spécifique de traduction de requêtes.

— Concevoir des prompts optimaux qui tiennent compte à la fois du langage de l'utilisateur et du contexte de la base de données.

Les résultats montrent des **améliorations significatives des performances** sur les exemples du dataset synthétique après fine-tuning, comparativement au modèle de base non adapté. :contentReference[oaicite :5]index=5

Cependant, l'auteur met aussi en évidence un compromis important : bien que le modèle affiné traduise mieux les requêtes proches du dataset d'entraînement, **sa capacité de généralisation vers des formulations très différentes est légèrement réduite**. :contentReference[oaicite :6]index=6

4. Évaluation et résultats

La thèse présente une évaluation empirique qui met en lumière plusieurs points clés :

— Le dataset synthétique est jugé **bien structuré et utile** pour entraîner des modèles à apprendre la tâche de traduction NL-to-NoSQL.

— Le processus de fine-tuning améliore notablement la capacité des LLMs à générer des requêtes NoSQL correctes à partir de requêtes en langage naturel.

— Une limite observée est la **légère perte de généralisation** lorsque les requêtes en entrée s'écartent du type d'exemples vus pendant l'entraînement. :contentReference[oaicite :7]index=7

L'auteur mentionne également que l'utilisation de modèles de plus grande taille pourrait améliorer encore les performances, mais cela nécessite des ressources mémoire et GPU plus importantes. :contentReference[oaicite :8]index=8

5. Contributions et perspectives

Cette thèse apporte plusieurs contributions notables à la recherche en interfaces de requêtes naturelles pour NoSQL :

- Une **méthode automatisée pour créer un dataset synthétique** adapté à la traduction de NL vers NoSQL.
- Une **expérimentation avec des LLMs affiné** sur ce dataset, démontrant des progrès dans la traduction de requêtes naturelles vers des requêtes NoSQL.
- Une discussion sur les limites actuelles des approches basées sur LLMs et les défis techniques associés, notamment en termes de ressources informatiques et de capacité de généralisation. :contentReference[oaicite :9]index=9

Pour l'avenir, l'auteur suggère d'explorer des techniques d'entraînement plus robustes et de tester des architectures de modèles plus grandes ou spécialisées pour encore mieux gérer la variété des requêtes naturelles. :contentReference[oaicite :10]index=10

6. Conclusion

La thèse de Alessandro Tola met en lumière une stratégie combinant création de données synthétiques et entraînement de grands modèles de langage pour rendre les bases NoSQL plus accessibles via des requêtes en langage naturel. Ce travail répond à un besoin réel dans le domaine des bases de données et ouvre des pistes intéressantes pour des interfaces plus conviviales et intelligentes. :contentReference[oaicite :11]index=11

7.Utilisation de l'article dans le projet

On a utilisé cet article dans le projet pour pouvoir convertir du SQL au NOSQL(Mongodb) la méthode consiste à traduire automatiquement des requêtes SQL en requêtes MongoDB en utilisant un dataset synthétique et des modèles de langage (LLMs). L'approche consiste d'abord à analyser la

requête SQL afin d'identifier ses clauses principales : `SELECT` (champs à récupérer), `FROM` (table cible), `WHERE` (conditions de filtre), `JOIN` (relations entre tables), `ORDER BY` et `GROUP BY`. Chaque clause est ensuite convertie en équivalent MongoDB : `SELECT` devient `$project`, `FROM` correspond à la collection, `WHERE` se transforme en `$match`, `JOIN` est remplacé par `$lookup` et éventuellement `$unwind`, `ORDER BY` par `$sort` et `GROUP BY` par `$group`. Le modèle de langage est entraîné sur des exemples synthétiques `(requête SQL, requête MongoDB)` pour apprendre à générer automatiquement le code MongoDB correct. Cette approche permet de maintenir la logique relationnelle d'origine tout en adaptant la structure document-oriented de MongoDB, garantissant que la requête convertie renvoie des résultats équivalents à la requête SQL initiale.