Datasets, métriques et LLMs pour la tâche Text-to-SQL

Mohamed Amine Kasmi, Wadie El amrani, Hiba L'Moudden, Fatima-Ezzahrae Broumi, Mohamed Hamza Ezzaki

Résumé Dans le contexte actuel de la gestion des données, la conversion du texte naturel en langage SQL représente un défi majeur, incarne l'intersection de la compréhension du langage naturel et de la science des données. Cet article propose une enquête et un état de l'art sur le problème de la conversion de texte en SQL. Nous présentons une analyse des ensembles de données, offrant une base pour évaluer les performances des modèles. De plus, nous discutons des métriques d'évaluation qui permettent de mesurer l'efficacité et la précision des approches de modélisation. En ce qui concerne les modèles, nous explorons les architectures qui facilitent la traduction du texte en requêtes SQL fonctionnelles. Enfin, nous proposons des perspectives pour la recherche dans ce domaine, soulignant les opportunités d'innovation et les défis à relever pour progresser vers une automatisation de l'extraction de données. Ce travail vise à servir de référence pour les futurs efforts de recherche et de développement dans la résolution du problème de texte en SQL.

Keywords: Text-to-SQL · Traitement automatique du langage naturel · Modèles de langage pré-entraînés · métriques · Benchmarks.

1 Introduction

Le Traitement Automatique du Langage Naturel (TALN) représente un domaine essentiel de l'intelligence artificielle, se concentrant sur la compréhension et la manipulation du langage humain par les machines. Au sein de ce vaste champ, la transformation de texte en SQL émerge comme une tâche cruciale, combinant les efforts des communautés du TALN et des bases de données. Cette transformation vise à convertir des questions formulées en langage naturel en requêtes SQL, permettant ainsi aux utilisateurs non techniques d'interroger efficacement des bases de données relationnelles. L'importance de cette transformation réside dans sa capacité à faciliter l'accès et l'analyse de vastes ensembles de données textuelles structurées. En effet, alors que les données textuelles deviennent de plus en plus omniprésentes dans de nombreux domaines, de la santé à l'éducation en passant par la finance, leur exploitation efficace nécessite souvent une compréhension et une manipulation à un niveau plus structuré. La transformation de texte en SQL permet de franchir cette étape critique en permettant aux utilisateurs de poser des questions en langage naturel et d'obtenir des réponses précises et pertinentes directement à partir des bases de données sous-jacentes. Cependant, malgré les avancées significatives dans ce domaine, plusieurs défis persistent. Ces défis incluent la généralisation des modèles de transformation de texte en SQL à travers différents domaines, la prise en charge de structures SQL complexes et l'optimisation de l'efficacité des requêtes générées. Pour relever ces défis et réaliser pleinement le potentiel de la transformation de texte en SQL, une évaluation approfondie et rigoureuse des techniques et des modèles existants est nécessaire. Dans cette optique, nous présentons une analyse des modèles et des méthodes utilisés dans la transformation de texte en SQL. Nous abordons les modèles de langage pré-entraînés , mettant en lumière leurs capacités spécifiques en matière de traitement du langage naturel. En outre, nous discutons des approches de traduction de texte en SQL, y compris les modèles comme Text-to-SQL et les techniques telles que le renforcement par imitation. Par la suite, nous examinons les différentes métriques utilisées pour évaluer la qualité des systèmes de transformation de texte en SQL. Nous comparons

également les méthodes d'évaluation humaine et automatique. Nous passons ensuite à la présentation des benchmarks, bases de données et jeux de données couramment utilisés dans ce domaine. Nous discutons des défis actuels et des directions futures de la recherche dans la transformation de texte en SQL.

2 Benchmarks et Datasets

Le domaine de la recherche text-to-SQL a été profondément influencé par l'introduction de benchmarks et d'ensembles de données complets qui défient et évaluent les capacités des modèles à travers différentes dimensions de complexité, de spécificité de domaine et de généralisabilité. Cette section passe en revue les principaux benchmarks et ensembles de données qui ont considérablement contribué à l'avancement des technologies text-to-SQL.

2.1 Spider

Spider a établi un nouveau standard pour les ensembles de données text-to-SQL en mettant l'accent sur des requêtes complexes et la généralisation inter-domaines [10]. Il a été conçu pour surmonter les limites des ensembles de données précédents qui se concentraient sur des scénarios mono-domaine avec des motifs de requête répétitifs. Avec 10 181 questions et 5 693 requêtes SQL uniques à travers 200 bases de données et 138 domaines, Spider défie les modèles non seulement de comprendre des constructions SQL complexes mais aussi de s'adapter à des schémas de base de données non vus. Spider est un jalon significatif pour évaluer la capacité des modèles à gérer la variabilité et la complexité du monde réel dans les tâches de requêtage de base de données.

2.2 UNITE

UNITE, créé par AWS AI Labs, élargit encore l'horizon des défis text-to-SQL en rassemblant 18 ensembles de données text-to-SQL disponibles publiquement en un seul benchmark unifié [4]. Cet assemblage résulte en un ensemble de données qui couvre plus de 12 domaines avec des requêtes SQL de plus de 3,9K motifs et inclut 29K bases de données, ce qui en fait la compilation la plus vaste de données text-to-SQL disponible à ce jour. La nature complète d'UNITE est conçue pour tester les modèles sur une large gamme de compétences.

2.3 CoSQL et SparC

CoSQL introduit la complexité du contexte conversationnel dans le parsing text-to-SQL [4]. Il simule des scénarios réels où les utilisateurs interagissent avec une base de données à travers une série de tours conversationnels, nécessitant des modèles capables de comprendre le contexte, de gérer les clarifications et d'affiner les requêtes basées sur les retours interactifs. SparC complète cela en se concentrant sur la réponse aux questions séquentielles. Les deux ensembles de données mettent l'accent sur la nécessité pour les modèles de maintenir le contexte au fil des tours conversationnels.

2.4 Divers ensembles de données dans UNITE

L'inclusion de divers ensembles de données dans UNITE, tels que WikiSQL, Criteria2SQL et d'autres, offre un paysage nuancé pour évaluer les modèles text-to-SQL [4]. Chaque ensemble de données au sein d'UNITE cible des aspects spécifiques de la tâche de traduction textto-SQL. Cette diversité permet une évaluation plus granulaire des capacités des modèles.
L'évaluation des modèles sur ces benchmarks révèle des défis significatifs pour atteindre une généralisation robuste et indépendante du domaine [4]. Bien que certains

Dataset	#Question	#SQL	#DB	#Domain	#Table
WikiSQL [12]	80654	77840	26521	1	773
Spider [10]	10181	45625	200	138	1020
ATIS [2]	5418	947	1	1	27
SParC [9]	4298	1276	200	138	200
CoSQL [11]	3007	15598	200	138	1020
BIRD [6]	10000	5000	300	-	600
UNITE [4]	15000	7000	400	-	800

Table 1. Aperçu des ensembles de données pour la conversion de texte en SQL par nombre de questions, de requêtes SQL, de bases de données, de tables SQL, et de domaines couverts.

modèles montrent des performances prometteuses sur des ensembles de données ou domaines spécifiques, les résultats globaux à travers divers benchmarks comme UNITE indiquent un écart considérable. Cela suggère que les modèles actuels ont encore du mal à saisir pleinement les subtilités du langage naturel. En résumé, ces benchmarks et ensembles de données représentent un effort collectif pour repousser les limites de ce que les modèles text-to-SQL peuvent réaliser. Ils posent les bases pour la prochaine génération de modèles capables de naviguer dans le spectre complet des défis présentés par les interfaces en langage naturel vers les bases de données.

3 Évaluations et métriques

Dans cette section, nous commençons par définir formellement le problème de l'analyse textuelle vers SQL. Ensuite, nous présentons les métriques d'évaluation officielles pour vérifier les analyseurs textuels vers SQL. Pour finir, nous détaillons les corpus de référence utilisés pour entraîner les analyseurs textuels vers SQL basés sur le deep learning.

3.1 Définition du problème

La conversion Texte-vers-SQL (T2S) a pour but de transformer une question en langage naturel (LN) en requête de langage structuré de requête SQL exécutable sur une base de données relationnelle. Comme le montre le Tableau 1, nous introduisons une notation formelle pour standardiser les définitions de tâches. En général,

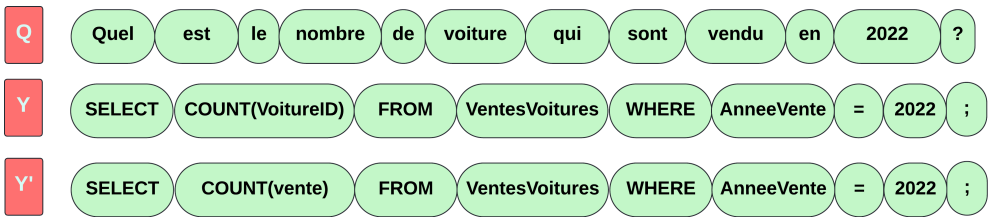


Figure 1. Comparaison de la requête SQL prédite Y' et réelle Y' en réponse à une question Q

les approches de conversion T2S existantes peuvent être catégorisées en configurations à *Tour unique* (*Single-turn*) indépendant du contexte et à *Multiples tours* (*Multi-turn*) dépendant du contexte. De manière formelle, pour la configuration à tour unique T2S, étant donnée une question Q et le schéma de base de données correspondant $S = \langle T, C \rangle$, notre objectif est de générer une requête SQL noté Y. Plus précisément, la question $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$ est une séquence de $|Q|$ jetons. Le schéma de base de données se compose de $|T|$ tables $T = \{t_1, t_2, \ldots, t_{|T|}\}$ et $|C|$

colonnes $C = \{c_1, c_2, \ldots, c_{|C|}\}$. Chaque table t_i est décrite par son nom contenant plusieurs mots $[t_{i,1}, t_{i,2}, \ldots, t_{i,|t_i|}]$. Chaque colonne $c^j_{t_i}$ dans la table t_i est représentée par des mots (une phrase) $[c^j_{t_i 1}, c^j_{t_i 2}, \ldots, c^j_{t_i |c^j_{t_i}|}]$. Nous notons l'ensemble de l'entrée par $X = \langle Q, S \rangle$.

Pour la configuration à multiples tours T2S, l'objectif est de convertir une séquence de questions LN en requêtes SQL correspondantes, où les questions LN peuvent contenir des ellipses et des anaphores faisant référence à des éléments antérieurs dans les questions LN précédentes. De manière formelle, soit $U = \{U_1, \ldots, U_T\}$ une séquence d'énoncés avec T tours, où $U_t = (X_t, Y_t)$ représente le t-ème énoncé qui est la combinaison d'une question LN X_i et d'une requête SQL Y_i. De plus, il y a un schéma de base de données correspondant S. Au t-ème tour, l'objectif de la conversion T2S à multiples tours est de produire la requête SQL Y_t conditionnée par la question LN actuelle X_t, les énoncés historiques $\{U_i\}_{i=1}^{t-1}$, et le schéma de base de données S.

3.2 Métriques

Dans cette section, nous proposons une analyse approfondie des métriques utilisés pour juger les performances des systèmes de conversion texte en SQL. Des études précédentes [7,3] ont déjà établi un cadre pour ces évaluations. Notre contribution s'enrichit du dernier état de l'art, intégrant de nouveaux benchmarks [4,6]. Nous allons au-delà en présentant une classification hiérarchique des indicateurs d'évaluation. Cette organisation facilite les comparaisons et met en lumière les caractéristiques distinctives des différentes approches. Grâce à cette stratégie, nous obtenons une vision plus précise des capacités actuelles et dirigeons efficacement les efforts de recherche vers le développement de solutions optimisées.

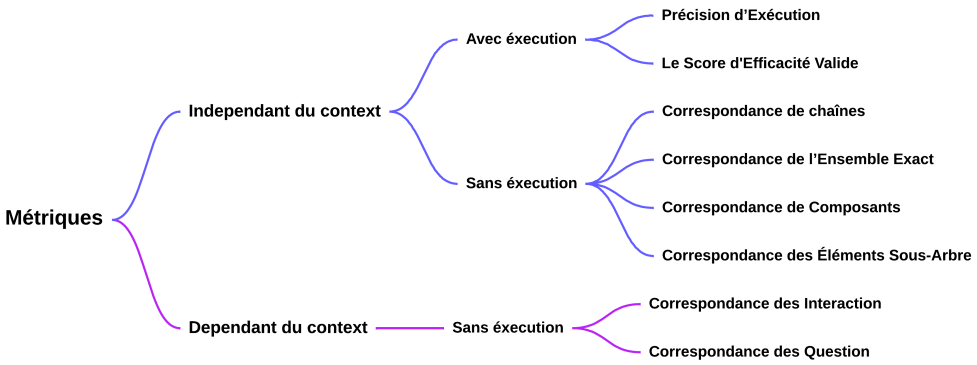


Figure 2. Classification hiérarchique des métriques d'évaluation de Text-to-SQL

Correspondance de chaînes (Logical Form Accuracy) [12], représente la méthode d'évaluation la plus élémentaire pour les systèmes de texte en SQL. Cette métrique traite les requêtes de vérité terrain et les requêtes prédites comme de simples chaînes de caractères et vérifie si elles sont identiques. Une correspondance est uniquement établie lorsque la requête prédite est exactement identique à la requête de vérité terrain, sans prendre en compte le fait que de nombreuses parties d'une requête SQL peuvent être écrites dans un ordre différent ou même de manière différente mais toujours équivalente, la figure 3 est une titre d'exemple.

$$\text{Correspondance de chaînes} = \begin{cases} 1 & \text{si } Q_{\text{vrai}} = Q_{\text{pred}} \\ 0 & \text{sinon} \end{cases} \tag{1}$$

Précision d'Exécution (Execution Accuracy) [10] illustré en figure 4 mesure l'aptitude d'une requête SQL générée à produire le bon résultat lorsqu'elle est exécutée

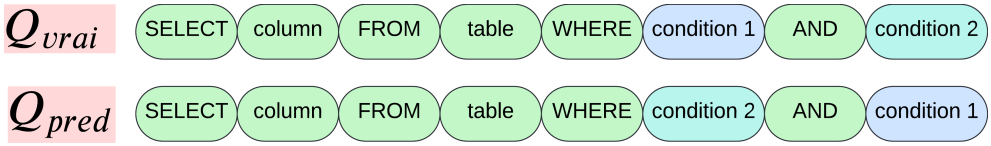


Figure 3. Exemple illustrant l'inefficacité de la métrique de correspondance de chaînes.

contre la base de données cible. Cette métrique évalue l'efficacité de la requête à retrouver les informations demandées. Considérant l'ensemble des résultats comme V_n exécuté par le SQL de vérité terrain n-ième Y_n, et l'ensemble des résultats $\hat{V}_n$ exécuté par le SQL prédit $\hat{Y}_n$, EX peut être calculé par :

$$EX = \frac{\sum_{n=1}^{N} 1(V_n, \hat{V}_n)}{N}, \tag{2}$$

où $1(\cdot)$ est une fonction indicatrice, qui peut être représentée comme :

$$1(V, \hat{V}) = \begin{cases} 1, & \text{si } V = \hat{V} \\ 0, & \text{si } V \neq \hat{V} \end{cases} \tag{3}$$

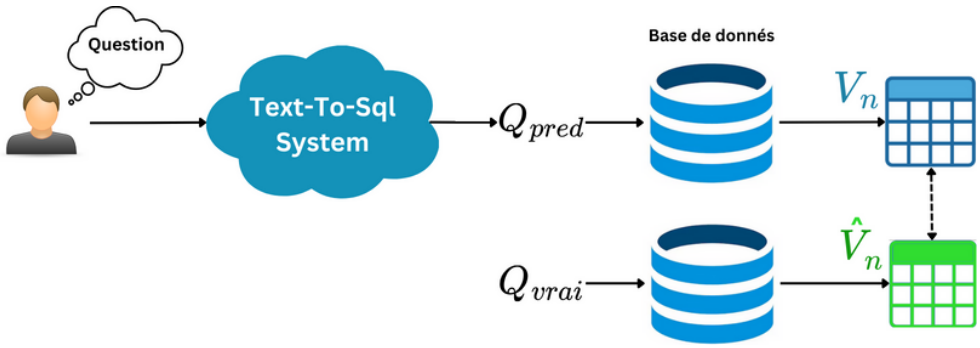


Figure 4. Illustration d'évaluation sur l'éxecution de la query

Correspondance de Composants (Component matching) [7,10] La métrique de évalue la précision d'une requête SQL prédite en examinant ses composants individuels. Cette évaluation se fait en décomposant la requête en parties distinctes : les clauses *SELECT*, *WHERE*, *GROUP BY* et *ORDER BY*. Pour chaque clause, on compare les sous-composants à ceux de la requête de référence. L'ordre des sous-composants dans chaque clause n'affecte pas cette évaluation. Ce qui importe, c'est la présence de tous les sous-composants nécessaires pour formaliser, soit C l'ensemble des clauses d'une requête SQL (par exemple, *SELECT*, *WHERE*, *GROUP BY*, *ORDER BY*). Pour chaque clause $c \in C$, nous décomposons la clause en sous-composants. Nous comparons ensuite les sous-composants de la requête prédite ($\hat{S}_c$) avec ceux de la requête de vérité terrain (S_c).

La correspondance des composants pour une clause c peut être définie comme :

$$\text{Match}_c = \begin{cases} 1, & \text{si } \hat{S}_c = S_c \\ 0, & \text{si } \hat{S}_c \neq S_c \end{cases} \tag{4}$$

La précision globale de la correspondance des composants de la requête est la moyenne des correspondances sur toutes les clauses, cette formule calcule la moyenne des correspondances exactes des sous-composants pour chaque clause de la requête,

$\mathbf{Q_{vrai}}$: Requête de Vérité Terrain	$\mathbf{Q_{pred}}$: Requête Prédite
SELECT	**SELECT**
— AuthorName	— AuthorId
— COUNT(*)	— COUNT(*)
WHERE	**WHERE**
— Year > 2000	— Genre = 'Science Fiction'
— Genre = 'Science Fiction'	— Year > 2000
GROUP BY	**GROUP BY**
— AuthorName	— AuthorId
ORDER BY	**ORDER BY**
— COUNT(*) DESC	— COUNT(*) DESC

Table 2. Illustration de la Correspondance de Composants (Component Matching)

sans tenir compte de leur ordre, pour illustrer, considérons les requêtes de vérité terrain et prédites fournies dans la table 2

$$\text{Précision} = \frac{\sum_{c \in C} \text{Match}_c}{|C|} \tag{5}$$

Précision de Correspondance de l'Ensemble Exact (Exact Set Match Accuracy) [10] est une métrique plus flexible car elle traite la requête SQL comme un ensemble d'éléments sans tenir compte de l'ordre spécifique de ces éléments. Par conséquent, elle va etre égale à 1 dans le cas d'exemple de la figure 3

Correspondance des Éléments de Sous-Arbres (Sub-tree Elements Matching) [1] est une expansion de *Component matching* [10] qui examine comment les différents composants ou segments d'une requête SQL générée, comme les clauses SELECT, WHERE, etc., correspondent à ceux de la requête de référence, permettant une certaine flexibilité dans l'ordre et la structure.

Score d'Efficacité Valide (Valid Efficiency Scor) [6] est conçu pour mesurer l'efficacité des SQL valides générés par les modèles. Toute requête SQL qui échoue à récupérer les bonnes valeurs sera déclarée invalide, car elles sont totalement inutiles si elles ne peuvent pas satisfaire les demandes de l'utilisateur, indépendamment de leur efficacité. Dans ce cas, la métrique considère à la fois l'efficacité et la précision des résultats d'exécution, fournissant une évaluation complète de la performance d'un modèle. Formellement, le SEV peut être exprimé comme :

$$\text{SEV} = \frac{\sum_{n=1}^{N} R(Y_n, \hat{Y}_n) \cdot (V_n, \hat{V}_n)}{N}, \quad R(Y_n, \hat{Y}_n) = \sqrt{\frac{E(Y_n)}{E(\hat{Y}_n)}} \tag{6}$$

où $R(\cdot)$ désigne l'efficacité d'exécution relative du SQL prédit par rapport au SQL de vérité terrain, permettant l'incertitude liée à l'état de la machine. $E(\cdot)$ est une fonction pour mesurer l'efficacité d'exécution absolue pour chaque SQL dans un environnement donné.

Précision de Correspondance des Questions (Question Match Accuracy). La Précision de Correspondance des Questions se concentre sur l'évaluation de la capacité du système à générer des requêtes SQL qui répondent correctement à la question en langage naturel, indépendamment de la structure exacte de la requête SQL.

Précision de Correspondance des Interactions (Interaction Match Accuracy) [2] [9]. La Précision de Correspondance des Interactions évalue la performance du système dans un contexte de dialogue ou de séquences de questions, en vérifiant si les requêtes SQL générées prennent en compte correctement le contexte et les informations fournies dans les interactions précédentes.

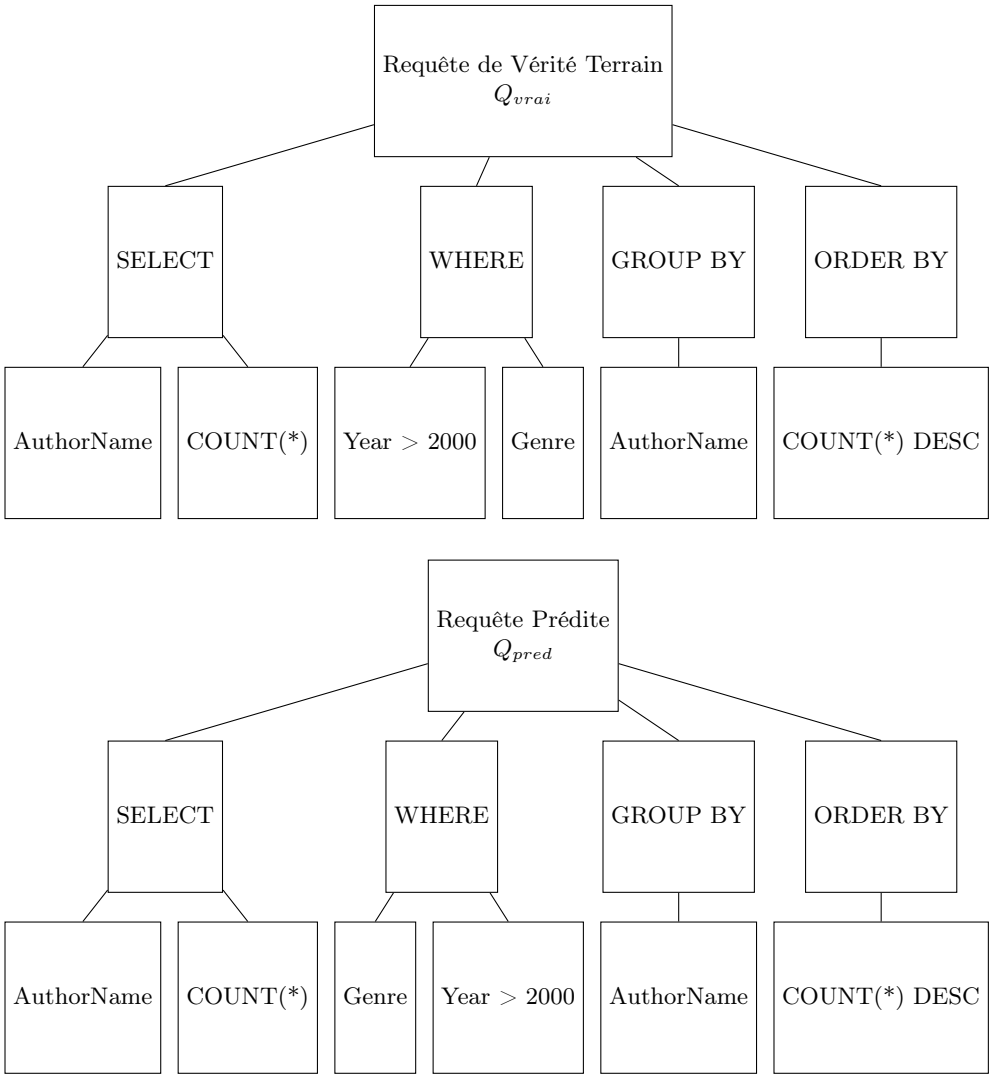


Figure 5. Correspondance des Éléments de Sous-Arbres (Sub-tree Elements Matching)

La métrique utilisée par chaque système dépend beaucoup du jeu de données pour lequel le système est créé et vise à se classer dans son classement. Plus précisément, les systèmes construits pour le jeu de données WikiSQL [12] utilisent la Précision de la Forme Logique et la Précision d'Exécution, tandis que ceux pour le jeu de données Spider [10] utilisent la Correspondance Exacte de l'Ensemble sans Valeurs et la Précision d'Exécution. Cela indique fortement l'influence que les créateurs de benchmarks ont sur la stratégie d'évaluation des systèmes texte-vers-SQL. Cela souligne également la responsabilité des futurs créateurs de benchmarks à résoudre les problèmes des métriques actuelles et à inclure des métriques d'évaluation plus approfondies.

4 Modèles

Cette section explore des modèles existants et les tentatives d'amélioration pour les correspondre à la tâche du Text-to-SQL. Elle présente leurs méthodologies d'apprentissage ainsi que les résultats accomplis.

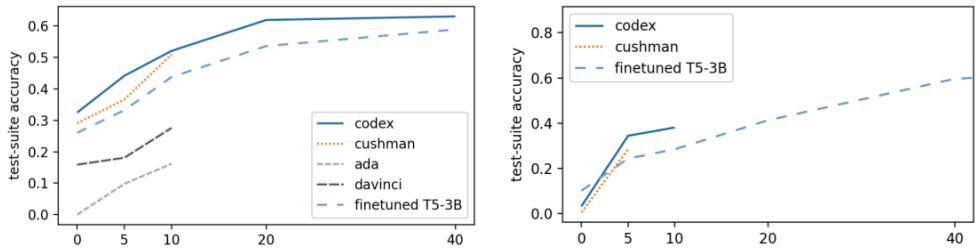
4.1 Codex vs T5-3B

Codex est un modèle de langage génératif, accessible via l'API d'OpenAI et affiné (*fine-tuned*) sur du code extrait de GitHub [8]. Son mode de fonctionnement repose sur la prédiction du jeton suivant lors de l'entraînement et de l'inférence.

Des études ont été menées pour savoir si ce modèle est concurrentiel sans avoir besoin d'être affiné sur des données spécifiques à la tâche du text-to-sql. Ces expérimentations ont utilisé les ensembles de données de GeoQuery et Scholar qui regroupent les exemples par modèle, c'est-à-dire deux exemples au même template ont la même structure de la requête SQL.

T5-3B a été choisit comme modèle de base pour effectuer les comparaisons, il est affiné sur Spider puis affiné davantage sur le nouveau domaine (GeoQuery ou Scholar). Le deuxième ensemble est le même inclus dans le prompt pour Codex.

Ce modèle a atteint 85.7% performance sur l'ensemble de test (*test-set performance*) après entrainement sur GeoQuery et 87.2% de Précision du test (*test accuracy*) quant à Scholar. A préciser que l'entrainement se déroule sur chaque base de données dans son intégralité. Cependant, avec un apprentissage à quelques exemples (*Few-Shot learning*) dans lequel un exemple aléatoire pour chacun des n templates les plus fréquents est retenu, Codex a pu le surpasser largement.



(a) T5 entraîné sur GeoQuery (b) T5 entraîné sur Scholar

Figure 6. Graphes de Test-suite accuracy. L'axe des abscisses est le nombre des exemples de few-shot [8]

Ces résultats montrent qu'en donnant quelques exemples d'un ensemble d'un domaine spécifique qui était utilisés pour affiner un modèle plus petit, comme prompt à Codex sera plus efficace.

4.2 Graphix-T5

En plus de comprendre la complexité sémantique et la variabilité dans la structure des requêtes, un autre grand défi persiste dans la tâche du text-to-SQL : la généralisation de domaine (*domain generalization*), c'est-à-dire la capacité à généraliser à un domaine de test jamais vu pendant l'entraînement. Le manque d'un modèle capable de relever ce défi a suscité une nouvelle idée.

T5, le modèle transformateur (*transformer*) spécialisé dans la tâche du text-to-text a atteint de hautes performances sur des benchmarks visant le même problème. Il va servir comme base qui va être augmentée avec d'autres composants afin de correspondre à la tâche du text-to-SQL tout en conservant cette propriété de généralisation [5].

Ce nouveau composant s'agit d'une couche appelée GRAPHIX-Layer, conçue d'abord pour intégrer les représentations sémantiques encodées par deux réseaux de neurones MHA et FFN stockés dans un block du transformateur, puis intégrer les représentations structurelles qui seront produites par un autre réseau appelé RGAT. A savoir que plusieurs couches GRAPHIX-Layer seront introduites et qu'à chaque fois la couche

s'occupe de faire la jointure des deux représentations par la formule :

$$\widetilde{H}_M^{(l)} = \widetilde{H}_S^{(l)} + \widetilde{E}_G^{(l)} \tag{7}$$

Cela donne la nouvelle architecture : Graphix-T5

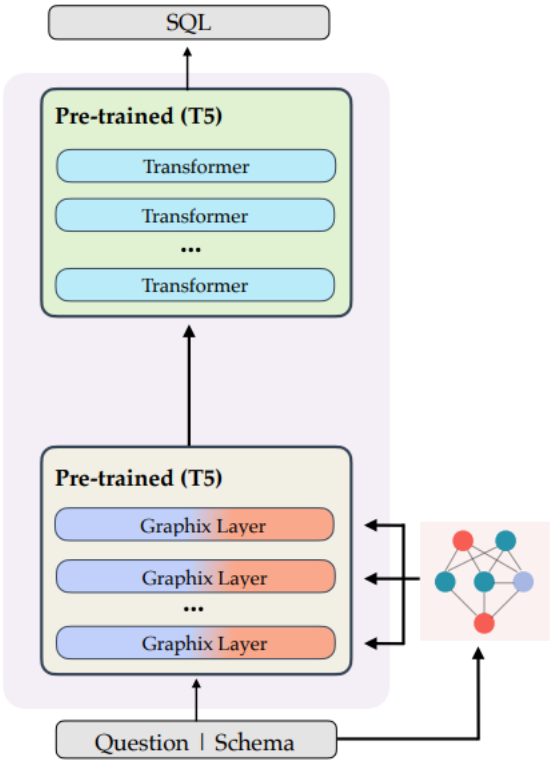


Figure 7. Architecture de Graphix-T5 [5]

Graphix-T5, testé sur quatre benchmarks (Spider, SYN, DK et REALISTIC), surpasse les performances d'autres variantes de T5 telles que Vanilla T5-large et T5-3B, même avec une configuration à faibles ressources. Il démontre une aptitude à gérer des cas complexes, plus proches de la réalité, et surtout, il affiche des capacités de généralisation augmentées.

5 Perspectives et défis futurs

5.1 Amélioration de la précision des modèles

Recherche sur les méthodes de correspondance avancées : Les recherches pourraient se concentrer sur le développement de techniques plus sophistiquées pour associer les candidats de la base de données aux éléments de requête. Cela pourrait inclure l'utilisation de réseaux neuronaux plus complexes, de techniques d'apprentissage multimodal ou de modèles de correspondance basés sur des graphes. Apprentissage à partir de données supplémentaires : L'utilisation de techniques d'apprentissage semisupervisé ou d'apprentissage par transfert à partir de données supplémentaires provenant de domaines similaires ou de bases de données différentes pourrait améliorer la précision des modèles. Exploration de l'incertitude : Les modèles de text-toSQL pourraient être

étendus pour fournir des estimations de l'incertitude associée à leurs prédictions, ce qui permettrait aux utilisateurs de mieux comprendre la fiabilité des résultats [7]

5.2 Gestion de la complexité des requêtes SQL générées

Développement de modèles multi-niveaux : Les modèles de text-to-SQL pourraient être conçus pour générer des requêtes SQL en plusieurs étapes, en décomposant les tâches complexes en sous-tâches plus gérables. Cela permettrait de mieux gérer la complexité des requêtes générées. Intégration de la connaissance du domaine : Les modèles pourraient être enrichis avec des connaissances sur le domaine spécifique de la base de données, ce qui les aiderait à générer des requêtes plus précises et plus efficaces.[7]

5.3 Intégration de la gestion des erreurs

Développement de mécanismes de correction d'erreurs : Les systèmes de text-to-SQL pourraient être dotés de mécanismes pour détecter et corriger les erreurs dans les requêtes utilisateur, en proposant des suggestions alternatives ou en demandant des clarifications. Apprentissage à partir des erreurs : Les modèles pourraient être entraînés sur des exemples d'erreurs courantes pour améliorer leur capacité à éviter les erreurs similaires à l'avenir. Extension à d'autres types de bases de données : Adaptation aux bases de données NoSQL et aux graphes de connaissances : Les recherches pourraient se concentrer sur le développement de modèles de text-to-SQL capables de travailler avec des bases de données non relationnelles ou basées sur des graphes, en tenant compte des spécificités de chaque type de base de données. Apprentissage à partir de données supplémentaires : Les modèles pourraient être entraînés sur des ensembles de données provenant de différentes sources pour améliorer leur capacité à généraliser à différents types de bases de données. [3]

5.4 Interprétabilité des modèles

Développement de techniques d'explication : Les recherches pourraient se concentrer sur le développement de méthodes pour expliquer les décisions des modèles de text-to-SQL, en mettant en évidence les parties importantes des entrées et des sorties du modèle. Utilisation de modèles interprétables : Les modèles de text-to-SQL pourraient être conçus de manière à être intrinsèquement interprétables, en utilisant des architectures qui permettent de comprendre facilement leur fonctionnement interne.

5.5 Adaptation aux langues et aux dialectes spécifiques

Collecte de données supplémentaires : Les recherches pourraient se concentrer sur la collecte de données supplémentaires dans différentes langues et dialectes pour entraîner des modèles de text-to-SQL adaptés à chaque langue spécifique. Adaptation des modèles aux particularités linguistiques : Les modèles pourraient être adaptés pour tenir compte des particularités linguistiques de chaque langue ou dialecte, en utilisant par exemple des techniques de prétraitement spécifiques à chaque langue.

6 Conclusion

En conclusion, cet article a présenté une exploration approfondie du domaine complexe de la transformation de texte en SQL dans le cadre du traitement automatique du langage naturel (TALN). En commençant par une introduction claire du domaine et de son importance dans le traitement des données textuelles, nous avons ensuite examiné en détail les modèles et les méthodes utilisés, mettant en lumière les modèles de langage pré-entraînés, ainsi que les approches de traduction telles que Text-to-SQL.

Nous avons également abordé les aspects d'évaluation et de métriques, en soulignant l'importance de méthodes d'évaluation fiables , ainsi que la comparaison entre les évaluations humaines et automatiques. De plus, nous avons présenté les benchmarks, bases de données et jeux de données utilisés couramment dans le domaine, fournissant ainsi des ressources précieuses pour la recherche future. Enfin, nous avons discuté des défis futurs et des perspectives de recherche, mettant en évidence la nécessité de continuer à améliorer la précision des modèles et de trouver des solutions pour gérer la complexité croissante des requêtes SQL générées. Dans l'ensemble, cet article illustre l'importance cruciale de la transformation de texte en SQL dans le domaine du TALN et met en lumière les avancées significatives ainsi que les défis persistants qui nécessitent une attention continue. En continuant à explorer ces avenues de recherche et en collaborant pour surmonter les obstacles, nous pouvons espérer ouvrir de nouvelles opportunités pour l'avenir du traitement automatique du langage naturel et de l'analyse de données textuelles.

Références

1. Hazoom, M., Malik, V., Bogin, B. : Text-to-sql in the wild : A naturally-occurring dataset based on stack exchange data. arXiv :2106.05006 [cs.CL] (2021). https://doi.org/10.48550/arXiv.2106.05006, nLP4Prog 2021
2. Hemphill, C.T., Godfrey, J.J., Doddington, G.R. : The atis spoken language systems pilot corpus. In : Proceedings of the workshop on Speech and Natural Language - HLT '90. pp. 96–101. Association for Computational Linguistics, Hidden Valley, Pennsylvania (1990). https://doi.org/10.3115/116580.116613
3. Katsogiannis-Meimarakis, G., Koutrika, G. : A survey on deep learning approaches for text-to-sql. The VLDB Journal **32**(4), 905–936 (Jul 2023). https://doi.org/10.1007/s00778-022-00776-8
4. Lan, W., et al. : Unite : A unified benchmark for text-to-sql evaluation. https://arxiv.org/abs/2305.16265 (Jul 2023). https://doi.org/10.48550/arXiv.2305.16265
5. Li, J., et al. : Graphix-t5 : Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. https://arxiv.org/abs/2301.07507 (Jan 2023). https://doi.org/10.48550/arXiv.2301.07507
6. Li, J., Hui, B., Qu, G., Yang, J., Li, B., Li, B., Wang, B., Qin, B., Cao, R., Geng, R., et al. : Can llm already serve as a database interface ? a big bench for large-scale database grounded text-to-sqls. https://arxiv.org/abs/2305.03111 (May 2023). https://doi.org/10.48550/arXiv.2305.03111, neurIPS 2023
7. Qin, B., et al. : A survey on text-to-sql parsing : Concepts, methods, and future directions. https://arxiv.org/abs/2208.13629 (Aug 2022). https://doi.org/10.48550/arXiv.2208.13629
8. Rajkumar, N., Li, R., Bahdanau, D. : Evaluating the text-to-sql capabilities of large language models. https://arxiv.org/abs/2204.00498 (Mar 2022). https://doi.org/10.48550/arXiv.2204.00498
9. Yu, T., et al. : Sparc : Cross-domain semantic parsing in context. https://arxiv.org/abs/1906.02285 (Jun 2019). https://doi.org/10.48550/arXiv.1906.02285
10. Yu, T., et al. : Spider : A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. https://arxiv.org/abs/1809.08887 (Feb 2019). https://doi.org/10.48550/arXiv.1809.08887
11. Yu, T., Zhang, R., Er, H.Y., Li, S., Xue, E., Pang, B., Lin, X.V., Tan, Y.C., Shi, T., Li, Z., Jiang, Y., Yasunaga, M., Shim, S., Chen, T., Fabbri, A., Li, Z., Chen, L., Zhang, Y., Dixit, S., Zhang, V., Xiong, C., Socher, R., Lasecki, W.S., Radev, D. : Cosql : A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. arXiv preprint arXiv :1909.05378 (2019)
12. Zhong, V., Xiong, C., Socher, R. : Seq2sql : Generating structured queries from natural language using reinforcement learning. https://arxiv.org/abs/1709.00103 (Nov 2017). https://doi.org/10.48550/arXiv.1709.00103