MultiTEND: A Multilingual Benchmark for Natural Language to NoSQL Query Translation

Zhiqian Qin[1], Yuanfeng Song[2*], Jinwei Lu[1], Yuanwei Song[3], Shuaimin Li[4], Chen Jason Zhang[1]

[1]The Hong Kong Polytechnic University, Hong Kong SAR, China
[2]AI Group, WeBank Co., Ltd, Shenzhen, China [3] Huawei, Hong Kong SAR, China
[4] Shenzhen Key Laboratory for High Performance Data Mining,
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences

Abstract

Natural language interfaces for NoSQL databases are increasingly vital in the big data era, enabling users to interact with complex, unstructured data without deep technical expertise. However, most recent advancements focus on English, leaving a gap for multilingual support. This paper introduces **MultiTEND**, the first and largest multilingual benchmark for natural language to NoSQL query generation, covering six languages: English, German, French, Russian, Japanese and Mandarin Chinese. Using MultiTEND, we analyze challenges in translating natural language to NoSQL queries across diverse linguistic structures, including lexical and syntactic differences. Experiments show that performance accuracy in both English and non-English settings remains relatively low, with a 4%-6% gap across scenarios like fine-tuned SLM, zero-shot LLM, and RAG for LLM. To address the aforementioned challenges, we introduce **MultiLink**, a novel framework that bridges the multilingual input to NoSQL query generation gap through a Parallel Linking Process. It breaks down the task into multiple steps, integrating parallel multilingual processing, Chain-of-Thought (CoT) reasoning, and Retrieval-Augmented Generation (RAG) to tackle lexical and structural challenges inherent in multilingual NoSQL generation. MultiLink shows enhancements in all metrics for every language against the top baseline, boosting execution accuracy by about 15% for English and averaging a 10% improvement for non-English languages.

1 Introduction

In the age of big data, NoSQL databases have become indispensable tools for managing vast amounts of unstructured and semi-structured data (Han et al., 2011). Unlike traditional relational databases, NoSQL databases offer more flexibility in schema design and can handle a wide variety of data types, making them particularly suitable for modern applications such as social media (Bhogal and Choksi, 2015), e-commerce (Nalla and Reddy, 2022), and real-time analytics (Ta et al., 2016). Nonetheless, the intricacy and heterogeneity of NoSQL query languages present a formidable challenge, especially for users who may not have advanced technical skills.

To address this challenge, the development of natural language interfaces (NLIs) for NoSQL databases has gained increasing attention. These interfaces are designed to allow users to interact with NoSQL databases in natural language, thus simplifying access to complex data and lowering the technical barriers. By translating Natural Language Queries (NLQs) into executable NoSQL queries (i.e., Text-to-NoSQL (Lu et al., 2025)), these systems can significantly enhance user productivity and data accessibility. However, existing natural language to NoSQL query generation systems and benchmarks have predominantly focused on the English language. This limitation severely restricts the usability of these systems for non-English speakers, who represent a significant portion of the global population.

To address above-mentioned issue, we introduce **MultiTEND**, the first multilingual benchmark for natural language to NoSQL query generation, covering six diverse languages: English, German, French, Russian, Japanese, and Mandarin Chinese (Sec. 3.1). MultiTEND not only expands the scope of natural language to NoSQL query generation to a multilingual context but also imposes additional challenges to the Text-to-NoSQL tasks. Based on the findings from our experiments (Sec. 4.2), We categorize the challenges in MultiTEND into Structural Challenge and Lexical Challenge. In particular, the Structural Challenge refers to difficulties models face in multilingual intention mapping tasks due to syntactic differences across languages, hindering accurate mapping to NoSQL operators. Additionally, the Lexical Challenge represents the schema linking difficulties models face in multilingual settings due to lexical differences (e.g., Japanese hiragana and katakana, Russian Cyrillic characters, and morphological variations in German and French) and the complexity of NoSQL structures (e.g., nested documents and array processing).

To tackle these challenges, we propose **MultiLink**, a novel framework designed to bridge the gap from multilingual input to NoSQL query generation. Specifically, MultiTEND extracts accurate operator sketches and relevant fields from multilingual NLQs through a parallel linking process, enabling the model to effectively generate high-quality NoSQL queries even

*Yuanfeng Song is the corresponding author.

Findings of the Association for Computational Linguistics: ACL 2025, pages 24632–24657
July 27 - August 1, 2025

in multilingual contexts. Through the incorporation of three novel and specifically designed components, namely Intention-aware Multilingual Data Augmentation, Parallel Multilingual Sketch-Schema Prediction, and Retrieval-Augmented Chain-of-Thought Query Prediction, MultiLink effectively generates high-quality NoSQL queries tailored for multilingual scenarios.

In summary, our contributions are as follows:

- We present MultiTEND, the largest multilingual benchmark for natural language to NoSQL query generation, which includes detailed construction methods and will be released to promote further research in this area.
- We conduct detailed analysis on the MultiTEND dataset, identifying the lexical and structural challenges in multilingual NoSQL generation, which arise from lexical and syntactic differences across languages as well as the inherent structural complexity of NoSQL queries.
- We introduce MultiLink, a novel framework designed for multilingual NoSQL query generation. By addressing both lexical and structural challenges through three innovative components, MultiLink achieves significantly better performance compared to other baselines in multilingual noSQL generation scenarios.
- We conduct extensive experiments on MultiTEND, demonstrating its challenging nature and the effectiveness of our proposed model in addressing these challenges.

The rest of this paper is structured as follows: Section 2 introduces the related work. Section 3 describes the construction process of the MultiTEND dataset. Section 4 describes the statistics and analysis of the MultiTEND dataset. Section 5 provides a detailed description of the MultiLink pipeline architecture. Section 6 presents the experimental setup and analysis of the results. Finally, Section 7 concludes the paper and discusses future directions. Section 8 discusses limitations.

2 Related Work

This study is closely related to the fields of Text-to-SQL and NoSQL Databases, as briefly surveyed below.

2.1 Text-to-SQL

Early research on Text-to-SQL primarily focused on meticulously designed rule-based methods, such as those in (Baik et al., 2020; Li and Jagadish, 2014a,b; Quamar et al., 2022; Sen et al., 2020), these methods used predefined rules or semantic parsers to translate NLQs into SQL but were inflexible and inadequate for handling increasingly complex database structures. With the rise of deep learning, the focus of Text-to-SQL research has gradually shifted towards methods that utilize deep neural networks, such as attention mechanisms (Liu et al., 2023) and graph-based encoding strategies (Hui et al., 2022; Li et al., 2023b; Qi et al., 2022; Wang et al., 2020; Xu et al., 2018; Zheng et al., 2022; Yu et al., 2021; Xiang et al., 2023). Alternatively, some approaches treat Text-to-SQL as a sequence-to-sequence problem by using encoder-decoder structured Pre-trained Language Models (PLMs) to generate SQL queries (Cai et al., 2018; Popescu et al., 2022; Qi et al., 2022).

In recent years, large language models (LLMs), which have demonstrated remarkable success across various domains, have also garnered increasing attention in the Text-to-SQL field (Dong et al., 2023; Gan et al., 2021; Gao et al., 2024; Li et al., 2023a; Lin et al., 2020; Pourreza and Rafiei, 2024; Qi et al., 2022; Rubin and Berant, 2021; Scholak et al., 2021). Current literature primarily focuses on two approaches with LLMs: prompt engineering and pretraining/fine-tuning. Prompt engineering methods typically involve using specific reasoning workflows which can be categorized into several reasoning modes, including Chain-of-Thought (CoT) (Wei et al., 2022) and its variants (Pourreza and Rafiei, 2024; Liu and Tan, 2023; Zhang et al., 2024b, 2023), Least-to-Most (Zhou et al., 2023; Gan et al., 2021; Arora et al., 2023), and Decomposition (Khot et al., 2023; Tai et al., 2023; Pourreza and Rafiei, 2024; Wang et al., 2025; Xie et al., 2024). To evaluate Text-to-SQL model performance in practical applications, several large-scale benchmark datasets have been developed and released, including WikiSQL (Zhong et al., 2017), Spider (Yu et al., 2018), KaggleDBQA (Lee et al., 2021), BIRD (Li et al., 2023c), and Bull (Zhang et al., 2024a) etc.

2.2 NoSQL Database

Traditional SQL databases face limitations with large-scale, unstructured, or semi-structured data in the internet and big data era, prompting the rise of NoSQL databases, which provide flexibility, scalability, and high performance in web applications and real-time data analysis (Moniruzzaman and Hossain, 2013). In the field of databases and NLP, current research primarily focuses on several key areas of NoSQL databases,including achieving scalability in data storage systems within large-scale user environments (Cattell, 2011), ensuring consistency in NoSQL databases (Diogo et al., 2019), addressing multi-tenant NoSQL data storage issues in cloud computing environments, particularly in scenarios involving resource and data sharing (Zeng, 2015), and realizing scalability, elasticity, and autonomy in database management systems (DBMS) within cloud computing environments (Agrawal et al., 2011).

Despite the extensive research on NoSQL across various domains, its accessibility remains a challenge, especially for non-expert users. Although Text-to-NoSQL tasks have been proposed to address this issue, existing NoSQL generation primarily supports English and overlooks the needs of non-English users. To tackle this issue, we introduce the Multilingual Text-to-NoSQL task, which is based on existing Text-to-NoSQL research and not only aims to reduce the barrier for non-expert users to utilize NoSQL databases by automatically convert-

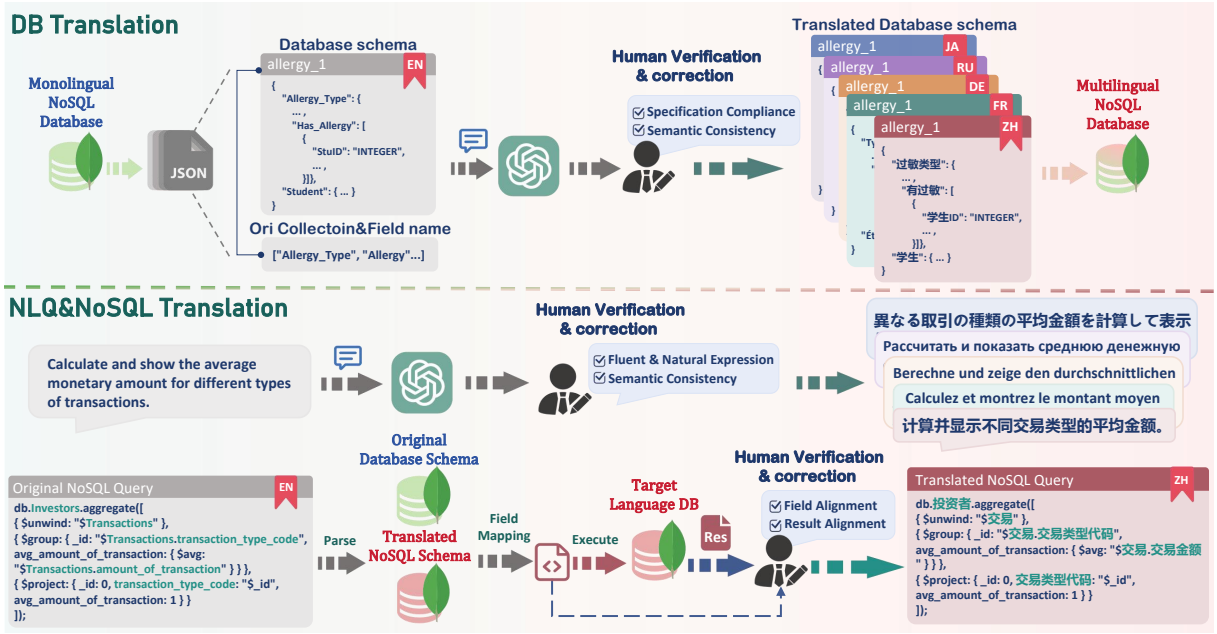


Figure 1: We developed a semi-automated pipeline to extend the monolingual dataset into a multilingual version through three steps: (1) Translation of Database Fields, where English-exclusive fields were translated using LLM-powered tools and manually verified; (2) Translation of NLQs, where NLQs were translated with few-shot prompting for semantic consistency and manually corrected; and (3) Translation of NoSQL Queries, where queries were programmatically parsed, updated with multilingual representations, and verified based on execution results. Each step combined machine-generated methods with rigorous manual verification.

ing NLQs into NoSQL queries but also addresses the gap in existing Text-to-NoSQL tasks that mainly support English while neglecting non-English users' needs. In this task, we also introduce MultiTEND, the largest multilingual benchmark for natural language to NoSQL query generation.

3 The MultiTEND Dataset

3.1 Overview

To address the limitation of existing datasets in the Text-to-NoSQL domain being solely constructed in English (Lu et al., 2025), we propose the first and the largest multilingual benchmark MultiTEND in this field, covering six languages: English, German, French, Russian, Japanese, and Mandarin Chinese. In this section, we'll introduce the dataset construction pipeline (Sec 3.2) and the manual correction processing (Sec 3.3).

3.2 Dataset Construction Pipeline

We segment the dataset's translation content into DB fields, NLQs, and NoSQL queries, employing a combination of prompt engineering (Sahoo et al., 2024) and manual corrections to construct the dataset.

Translation of DB Fields We interpret the task of translating database fields as obtaining relationship maps from English to five different languages for the database fields. We encapsulate instructions and contextual information conducive to accurate translation, such as the database schema, the fields to be translated, and the required output format into prompts (As shown in Appendix E.1) and utilize a large language model (LLM) to complete the translation process. The translation results undergo detailed human inspection and correction (as shown in Section 3.3), ultimately yielding five maps from English to each target language for every database. These well-checked maps are used for the translation of the databases, resulting in a total of 924 databases covering six languages, derived from the original 154 English-language databases (Figure 1).

Translation of NLQs We have established the following requirements for the translation of the NLQs: (i) Semantic alignment; (ii) Preservation of specific referenced values; (iii) Fluency in language expression. Among these, the requirement to preserve specific referenced values corresponds to multilingual database fields, where the actual database values remain consistent with the original English database to ensure data consistency. To achieve efficient and high-quality NLQ translation, we have designed a step-by-step, query-intent-based, structured Prompt with contextual examples for multilingual NLQ translation (Appendix E.1). By encouraging the model to think step-by-step, we ensure the fluency and accuracy of the translation. Finally, we perform manual verification and correction of the generated NLQ (Sec 3.3) to ensure that the translated NLQ meets the specified requirements.

Translation of NoSQL Queries As mentioned earlier, for each database, we have already obtained five mapping tables from English to target language for the field names (Figure 1) and conducted manual reviews

Dataset	Input	Output	Query Source	Query Size	Languages
Spider (Yu et al., 2018)	NLQ	SQL Query	Human-Labeled	5693	English
nvBench (Luo et al., 2021a)	NLQ	Data Vis Query	Rule-based Synthesised	7247	English
OverpassQL (Staniek et al., 2024)	NLQ	Spatial Query	Crowdsourcing collected	3890	English
TEND (Lu et al., 2025)	NLQ	NoSQL Query	Machine generated ✓	3404	English
MultiTEND	NLQ	NoSQL Query	Machine generated & Human Check ✓	20351	Multiple(6) ✓

Table 1: Comparison of MultiTEND with Other Existing Benchmarks in Natural Language Interface fields.

and corrections. Based on these well-reviewed mapping tables, we mapped the fields referenced in each NoSQL query from English to five different languages. For each translated NoSQL query, we first filter out incorrect queries by executing them and checking the execution results, then apply manual corrections to strictly ensure the accuracy of the translated NoSQL queries.

3.3 Manual Correction

Typical Errors Analysis As mentioned in Sec 3.2, we completed the translation of db fields, NLQs, and NoSQL queries empowered by LLM and conducted final manual inspections and corrections. We summarize some of the typical mistakes discovered during the inspection of translation process as follows. For the translation part of DB fields, we identified two typical error categories: Polysemy and Abbreviation. **Polysemy**: Some fields can have different meanings depending on the database scenario, which is one reason for the inappropriate translation of certain database fields. For example, the term 'Movements' in the 'Aircraft_Movements' field could refer to 'motion,' 'movement,' or more specifically, 'take-offs and landings.' By analyzing the data type and specific values of this field, it becomes evident that 'take-offs and landings' is the most suitable meaning within the context of aircraft operations. **Abbreviation**: Translating abbreviations in database fields, taking into account the database context, is inherently a challenging task. Such errors constituted a larger proportion of the issues we detected in the translation of db fields. For example, 'fname' might be incorrectly translated as 'f姓名' whereas it should be translated as '名' when considering its neighbor 'lname'; similarly, 'f_id' could mean 'flight ID' or 'file ID,' depending on the theme of the collection. Similarly, 'HS' from the 'soccer_2' database could stand for 'High School,' 'Home State,' or 'Historical Score.' However, upon examining the neighboring fields and specific values within the collection, it turns out that 'HS' actually means 'Historical Score.' In the translation part of NLQs, we found that most of the NLQs not meeting the requirements mentioned in Sec 3.2 were due to insufficient fluency in the language, such as translating "How many papers are 'Atsushi Ohori' the author of?" into "有多少论文是'Atsushi Ohori'的作者?". This result comes from directly translating 'are' and 'of' without considering the overall structure of the sentence.

Correction Criteria Based on the error cases observed during the aforementioned manual inspection process, we made several adjustments to the dataset aiming to ensure the following aspects: DB fields adhere to standard database design rules and feature precise translations of polysemous words and abbreviations, aligning them with the context of the database; NLQs maintain semantic consistency and are expressed fluently and naturally; and NoSQL query results are fully consistent with the original queries. For example, we carefully examined some abbreviated fields in the original database to ensure that these abbreviations, which are difficult to understand without context, accurately convey their original meanings after translation (e.g., "flno" remains equivalent to "flight number" after translation). Additionally, we paid special attention to potential field duplication issues after translation, particularly for fields originally distinguished by case or singular/plural forms, as such differences might result in identical expressions in the target language. For example, the collection name "continents" and the field name "Continent" might both be translated as "洲" in Chinese; similarly, the collection name "city" and the field name "City" might both be translated as "都市" in Japanese.

4 Dataset Statistics and Analysis

4.1 Statistics of MultiTEND

After our multilingual extension of TEND (Lu et al., 2025), MultiTEND ultimately includes a total of 154 databases with different content, comprising 924 databases in total, and 101,789 (NLQ, NoSQL) pairs (including 20,351 distinct NoSQL queries). The count of 101,789 pairs is derived from the fact that each query corresponds to five NLQs, with each NLQ further represented in six language versions. Approximately 16.6% of all NoSQL queries use the find method (which includes filter, projection, sort, and limit operations), while the remaining queries use the aggregate method (implemented through pipelines, which include but are not limited to project, group, match, sort, limit, lookup, and count stages) (Detailed statistics of MultiTEND see Apendix A.1). Compared to other well-regarded datasets in different fields, such as Spider (Yu et al., 2018), NvBench (Luo et al., 2021b), OverpassQL (Staniek et al., 2024), and TEND (Lu et al., 2025), MultiTEND stands out for its vast scale and comprehensive multilingual coverage (as shown in Table 1). In terms of scale, MultiTEND boasts 101,789 NLQs and 20,351 corresponding queries, far surpassing other datasets like Spider, which has 10,181 NLQs and 5,693 queries; NvBench, with 25,750 NLQs and 7,247 queries; and TEND, featuring 17,020 NLQs and 3,404 queries. Re-

Metric	Model	EN	ZH	FR	DE	JA	RU	AVG (5 langs)
EM	Fine-tuned Llama	17.05%	13.57%	16.53%	15.78%	16.40%	14.51%	15.36%
	Zero-shot LLM	0.29%	0.61%	0.61%	0.54%	0.54%	0.29%	0.52%
	RAG for LLM	16.09%	13.98%	15.62%	14.33%	12.02%	13.89%	13.97%
	SMART	18.85%	13.94%	18.38%	18.30%	18.05%	15.89%	16.91%
EX	Fine-tuned Llama	44.61%	36.86%	41.26%	41.44%	43.32%	38.23%	40.22%
	Zero-shot LLM	36.58%	28.99%	33.86%	34.91%	30.63%	29.68%	31.61%
	RAG for LLM	51.70%	47.02%	49.28%	48.59%	45.12%	45.99%	47.20%
	SMART	48.86%	38.05%	44.69%	44.22%	43.30%	41.03%	42.26%

Table 2: Comparison of Exact Match and Execution Accuracy results for each model across different languages on MultiTEND. Notice that **AVG** is the average value of the corresponding metric across the 5 **non-English** languages

garding multilingual support, unlike other datasets that primarily offer data in English, MultiTEND supports six distinct languages, greatly broadening its applicability and research value. Additionally, MultiTEND's semi-automated construction process, which combines machine-generated data with manual verification, provides significant advantages in terms of scalability and efficiency during its development.

4.2 Analysis and Findings

To clarify the challenges posed by multilingual Text-to-NoSQL tasks for existing models, we conducted a series of experiments and derived key findings from the analysis of the experimental results (See Appendix A.2). Based on the findings, we categorize the challenges in MultiTEND into **Structural Challenge** and **Lexical Challenge**. (i) The **Structural Challenge** refers to the difficulties models face in performing intention mapping tasks in multilingual contexts, primarily due to significant syntactic differences across languages, which reduce the model's ability to understand and parse user intentions, making it harder to accurately map them to corresponding NoSQL operators. (ii) The **Lexical Challenge** refers to the difficulties models encounter in schema linking in multilingual environments, mainly stemming from lexical differences (e.g., Japanese hiragana and katakana, Russian Cyrillic characters, and morphological variations in German and French) and the complexity of NoSQL structures (e.g., nested documents and array processing). These factors collectively increase the model's comprehension difficulty, leading to a significant decline in mapping accuracy.

5 Method

To address the complex challenge of generating NoSQL queries from multilingual NLQs, we introduce the innovative MultiLink framework. This framework leverages a problem decomposition strategy and an efficient Parallel Linking Process to effectively address the challenges of multilingual NoSQL query generation. In this chapter, we provide a comprehensive overview of MultiLink.

5.1 Overview

As shown in Figure 2 and Algorithm 1, MultiLink is primarily divided into three key components: Intention-aware Multilingual Data Augmentation, Parallel Multilingual Sketch-Schema Predictor (including a NoSQL Sketch Generator and a Schema Linking Generator), and Retrieval-Augmented Chain-of-Thought Generator. By leveraging fine-tuned SLM (Small Language Model) technology, which combines low computational resource consumption, shorter training cycles, and sufficient performance to meet our task demands, and integrating it with our data augmentation approach,we achieve cost-effective and high-yield prediction of intention mapping and schema linking. This method specifically targets the extraction of lexical and structural challenges from multilingual NLQs and effectively empowers LLM by providing contextual information, enabling the generation of more accurate and reliable NoSQL queries without significantly exacerbating model hallucinations. To address these challenges, we employ English, a high-resource language, as a unified bridge for conveying operator information across languages, while utilizing the corresponding languages for schema linking to maintain the model's sensitivity to relevant fields in each language. Finally, through our efficient RAG (Retrieval-Augmented Generation (Lewis et al., 2020)) retrieval technique and a Chain-of-Thought prompting strategy, we consolidate the extracted information into structured contexts. This approach not only mitigates hallucinations in LLMs but also significantly enhances the accuracy of NoSQL generation in multilingual environments.

5.2 Intention-aware Multilingual Data Augmentation (MIND)

We augment the training data using a Intention-aware Chain-of-Thought (CoT) (Wei et al., 2022) guided multilingual query augmentation strategy. Given the original (NLQ, NoSQL) pairs from the TEND dataset, we employ a LLM to synthesize additional pairs with diverse querying intents(for detailed prompt example see E.2). The augmentation process involves the following steps: (1) analyzing the structural relationships between col-

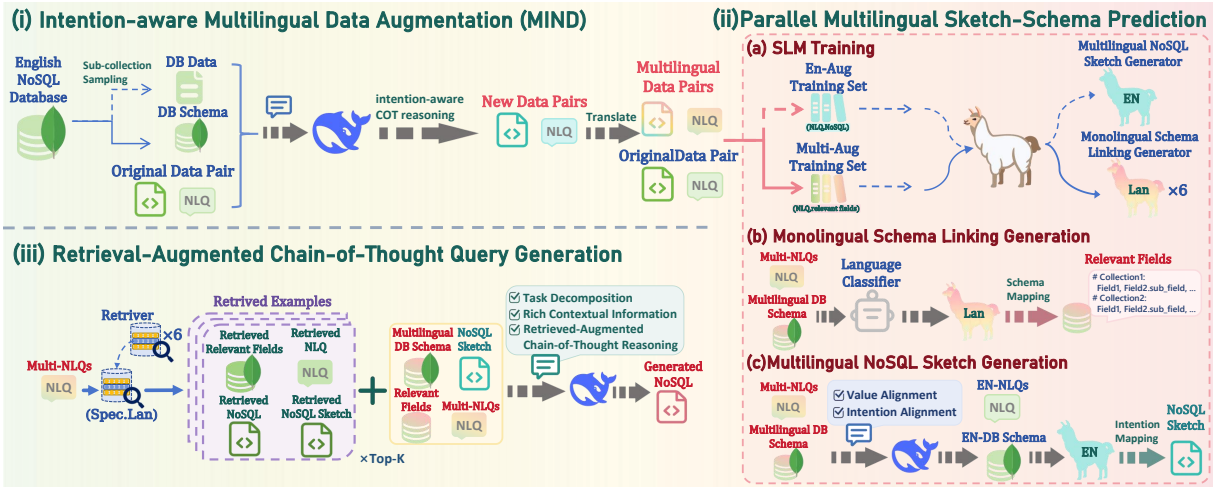


Figure 2: The pipeline of our proposed MultiLink method. MultiLink consists of three main process: (i) Intention-aware Multilingual Data Augmentation (MIND), which enriches training data by generating diverse query pairs through multilingual synthesis and comprehensive database schema analysis; (ii) Parallel Multilingual Sketch-Schema Prediction, which parallelly maps multilingual NLQs intentions to operators and entity mentions to schema elements,including: (a) Multilingual NoSQL Sketch Generation, which generates intermediate NoSQL sketches reflecting operator mappings; (b) Monolingual Schema Linking Generation, which performs precise schema linking for each language;(iii) Retrieval-Augmented Chain-of-Thought Query Prediction, which synthesizes the final NoSQL query by integrating operator and schema mappings with multilingual context.

lections and fields in the MongoDB schema; (2) identifying logical relationships between fields and collections based on the NLQ and the referenced schema portions in the NoSQL query; (3) generating new NoSQL queries with completely different intents from the original queries; (4) creating NLQs that match the intents of the generated NoSQL queries and expanding them into paraphrased variants; and (5) synthesizing corresponding NLQs in multiple languages (e.g., German, French, Russian, Japanese, and Mandarin Chinese). This process not only increases the diversity of multilingual training data but also enhances the performance of the SLMs in intention mapping and schema linking, ensuring that our pipeline can effectively handle multilingual inputs.

5.3 Parallel Multilingual Sketch-Schema Predictor

The Parallel Multilingual Sketch-Schema Predictor is a key component of our pipeline, designed to address the lexical and structural challenges in multilingual NoSQL generation in parallel. The predictor consists of two parallel submodules: (i) the Multilingual NoSQL Sketch Generator, which maps multilingual NLQs intents to NoSQL operators via a unified intermediate representation (i.e., English); and (ii) the Monolingual Schema Linking Generator , which maps entity mentions in the NLQ to the corresponding schema elements in the target database. By executing these submodules in parallel, Sketch-Schema Predictor ensures high accuracy and efficiency in both operator and schema mapping across multilingual contexts.

Multilingual NoSQL Sketch Generator To address the challenge in intention mapping exacerbated by lexical diversity and syntactic heterogeneity in multilingual contexts, we designed Multilingual NoSQL Sketch Generator, a sketch generator incorporating the mapping from intention to operator. We adopt English, a high-resource language, as a unified bridge for cross-lingual transfer of operator information. Given a multilingual NLQ, Sketch Generator, leveraging the power of an LLM, extracts and anchors the underlying query intent by translating both the NLQ and the database schema into English. The translated English NLQ, along with the corresponding database schema, is then fed into the fine-tuned SLM to generate an intermediate NoSQL sketch. This sketch reflects operator mappings (e.g., sort, filter, aunwind), but does not include precise schema field references. By unifying multilingual intentions into a single language (i.e., English), Sketch Generator efficiently and cost-effectively streamlines the operator mapping process and ensures consistency across languages.

Monolingual Schema Linking Generator In the Multilingual Text-to-NoSQL task, models are typically required to have a thorough understanding of entity mentions across complex lexicons in different languages (e.g., hiragana and katakana in Japanese, Cyrillic characters in Russian, and the rich and varied lexical forms in German and French). At the same time, they must possess the ability to cross-linguistically map entity mentions in NLQ to the corresponding fields in the database schema. The nested and unstructured nature of NoSQL schemas further exacerbates the challenge in schema linking. Therefore, we designed an efficient format to express schema linking results, such as '# Collection1: Field1, Field2.sub_field,... \n # Collec-

Algorithm 1: The MultiLink Algorithm

```
Inputs  : NLQ list in Test Set 𝒬;
          DB schema list in Test Set 𝒟;
          DB list in Training Set 𝒟′;
          NLQ list in Training Set 𝒬′;
          NoSQL list in Training Set 𝒩′;
Output  : NoSQL list 𝒩
Procedure MultiLink(𝒬, 𝒟):
    // Data Augmentation
    𝒬′_aug, 𝒩′_aug, 𝒮′_aug ← AugmentData(𝒬′,𝒩′,𝒟′)
    // Sketch SLM Fine-tuning
    ℳ_n ← TrainSLM(SLM, 𝒬′_aug_en, 𝒩′_aug_en)
    // Schema Linking SLM Fine-tuning
    ℳ_s ← TrainSLM(SLM, 𝒬′_aug, 𝒮′_aug)
    // Build Vector Library
    𝒱 ← BuildVecLib(𝒬′, 𝒩′)
    // Pipeline of MultiLink
    𝒩 ← []
    for each (q, d) ∈ (𝒬, 𝒟) do
        // Language Classification
        L ← LangClassify(q)
        // Translation
        q_en, d_en ← Translate(q, d)
        // Sketch Prediction
        n_sk ← SLMPredict(ℳ_n, q_en, d_en)
        // Schema Linking Prediction
        s ← SLMPredict(ℳ_s, q, d)
        // Retrieval-Aug CoT Generation
        ℰ ← Retrieve(q,𝒱,L)
        n_gen ← Generate(q, d,n_sk, s,ℰ)
        𝒩.append(n_gen)
    end
    return 𝒩
```

tion2:..' (e.g., '# 产品: 产品价格, 投诉.员工ID\n# 员工: 员工ID'). Based on this format, we constructed corresponding corpora for schema linking in different languages. Combined with a language classifier, the schema linking generator inputs the multilingual NLQs (e.g., "登山者为位于乌干达的山峰记录的攀登时间是什么？") into a fine-tuned SLM (Schema Linking Model) trained on language-specific schema linking corpora, in order to accurately map the entity mentions in the NLQ to the corresponding schema elements in the target database(e.g., # 山脉: 国家, 登山者, 登山者.时间). By employing separately fine-tuned SLMs for schema linking in each language, Schema Linking Generator ensures high accuracy in schema linking result, addressing the lexical challenges in multilingual NoSQL generation.

5.4 Retrieval-Augmented Chain-of-Thought Query Generator

The final module of our pipeline is the Retrieval-Augmented Chain-of-Thought Generator, which synthesizes the final NoSQL query by integrating the results from the previous steps. Given a multilingual NLQ, the Query Generator include: (i) the reference English NoSQL query with operator mappings (from Sketch Generator); (ii) the database schemas; (iii) the schema linking result of current NLQ (from Schema Linking Generator); and (iv) retrieved examples from the corresponding language example library created from the training data. Using a Retrieval-Augmented Chain-of-Thought reasoning approach, the Query Generator significantly enhances the reasoning capabilities of the LLM by referencing similar retrieved examples in the same language and guiding the query generation process step-by-step. By combining the results from Sketch Generator and Schema Linking Generator, the Query Generator addresses the inherent challenges of multilingual scenarios. Leveraging the enhanced reasoning capabilities of the LLM, Query Generator accurately synthesizes and utilizes key contextual information, generating precise and semantically consistent NoSQL queries with higher scores and better performance compared to baseline models.

6 Experiments and Analysis

6.1 Experimental Setup

Dataset We conducted cross-domain partitioning of the MultiTEND dataset for different languages, ensuring that the training set for each language contained the same sample content. The dataset was divided into original language-specific training and test sets at a ratio of 0.85:0.15, and the training and test sets for each language were merged to form a multilingual training set and test set encompassing six languages. For the additional dataset obtained through data augmentation, which contains 2,666 different NoSQL queries with 5 NLQs corresponding to each NoSQL query in each language, we directly added it to the original training set to create an augmented language-specific training set. By combining these datasets for each language, we obtained a multi-augmented training set covering six languages. Comprehensive statistics on dataset splits, including distinct training sets used across MultiLink modules, are provided in the Appendix A.1.

Baselines We utilized a variety of popular neural network models (i.e., LLM-based prompting methods (i.e., **Zero-shot LLM**, **Few-shot LLM**, **RAG for LLM**), SLM-based fine-tuning methods (i.e., **Fine-tuned SLM**) and existing Text-to-NoSQL methods (i.e., **SMART** (Lu et al., 2025)) as baseline models for a comprehensive performance comparison with MultiLink. The details of these baseline models can be found in Section C.1 of the Appendix.

Evaluation Metrics Following other text-to-NoSQL study like SMART (Lu et al., 2025), we report results using the same metrics including *Exact Match* (EM) and *Execution Accuracy* (EX), each with more detailed subdivisions such as *Query Stages Match* (QSM) and *Query Fields Coverage* (QFC) under EM, and *Execution Fields Match* (EFM) and *Execution Value Match* (EVM) under EX. The detailed definition of these metrics can be found in Section C.2 of the Appendix.

Implementation Details The SLM used in MultiLink is "Llama-3.2-1B", with a full-parameter fine-

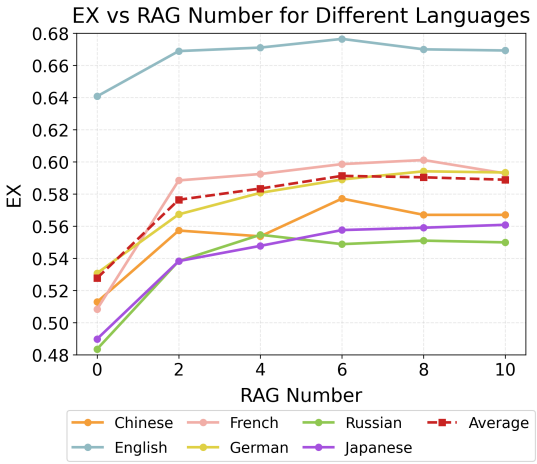
Model	EM	QSM	QFC
Fine-tuned Llama	15.64%	55.90%	58.44%
Zero-shot LLM	0.48%	49.35%	59.36%
Few-shot LLM	10.82%	56.17%	61.12%
RAG for LLM	14.32%	59.79%	67.19%
SMART	17.23%	59.12%	61.94%
MultiLink (Ours)	**25.54%**	**64.01%**	**73.17%**

(a) Query-based Metric Results (Avg of 6 langs)

Model	EX	EFM	EVM
Fine-tuned Llama	40.95%	81.41%	70.46%
Zero-shot LLM	32.44%	55.98%	59.22%
Few-shot LLM	36.69%	64.43%	64.71%
RAG for LLM	47.95%	74.22%	69.80%
SMART	43.36%	85.05%	**76.37%**
MultiLink (Ours)	**59.12%**	**85.66%**	74.01%

(b) Execution-based Metric Results (Avg of 6 langs)

Table 3: Overall Performance Metrics



(a) Execution-based Metric

Figure 3: Parameter study

tuning strategy and set to 3 epochs. The LLM used is "DeepSeek-V3", with the parameter setting 'temperature = 0.0'. The text-to-embedding model used is "text-embedding-ada-002".

6.2 Performance Comparison

Table 3 presents the average performance of MultiLink and baseline methods across all languages in MultiTEND (For detailed per-language and per-metric analysis of MultiLink and all baselines, please refer to the Appendix D.1). As shown in Table 3, the fine-tuned Llama (Dubey et al., 2024) and LLM-based methods (Zero/Few-shot LLM, RAG for LLM) underperform below 50%, with Zero-shot LLM systematically exhibiting the weakest results due to deficiencies in query intent comprehension and critical failures in processing nested arrays/multi-set associations. While the approach of RAG for LLM with enhanced contextual information shows relatively better performance in the Execution Accuracy (EX) metric that directly reflects query execution outcomes. This suggests that neither pure fine-tuning nor direct reliance on LLM capabilities constitutes an effective solution for multilingual NoSQL challenges.

Additionally, while SMART, which is designed for English contexts, performs averagely in multilingual NoSQL tasks, showing significantly different results between English and non-English languages (see Table 2). This indicates that existing Text-to-NoSQL systems, cannot be directly extended to non-English scenarios. In contrast, our proposed MultiLink framework demonstrates superior performance in multilingual environments, outperforming all existing models across every metric. Particularly noteworthy is its 11% absolute improvement over the best-performing baseline in the crucial EX metric. These results validate that MultiLink's design effectively addresses multilingual NoSQL generation challenges and produces high-quality queries.Due to space limitations, please refer to appendix D.1 for more detailed analysis.

6.3 Parameter Study

To explore the performance of MultiLink under different parameters, we conducted a hyperparameter experiment on the number of retrieved examples (RAG num) using the test set of the MultiTEND dataset. As shown in Figure 3, the figures illustrate the execution accuracy (EX) of MultiLink under different RAG numbers across multiple languages (for complete metric results of the parameter study, see the appendix D.2). As the RAG number increases, MultiLink exhibits slight fluctuations in various metrics across different languages. The execution accuracy (EX) shows that English performs the best, with minor fluctuations around 67%; French and German are in the middle range, at 5860%; while Chinese, Japanese, and Russian are lower, consistently staying within the 54%-57% range. The average performance across all languages (represented by the red dashed line) shows a steady increase and stabilizes after reaching a RAG number of 6. This indicates that as the RAG number increases, the model's performance improves within a certain range, and MultiTEND achieves its best performance at a RAG number of 6.

6.4 Ablation Study

In this section, we conduct an ablation study to examine the contribution of each module in MultiLink. We first measure the results of MultiLink based on the complete pipeline designed. Then, we evaluate the contribution of each module by removing different key modules from MultiLink, specifically: (i) removing the Sketch Generator (w/o Sk-G); (ii) removing the Schema Linking Generator (w/o SL-G); (iii) using only the Retrieval-Augmented Chain-of-Thought Generator (only GEN); and (iv) using components without data augmentation (w/o AUG).

Additionally, we include the experimental results of

Method	EM	QSM	QFC
Few-Shot LLM	10.82%	56.17%	61.12%
RAG for LLM	14.32%	59.79%	67.19%
MultiLink (Ours)	25.54%	64.01%	73.17%
- w/o Sk-G	25.51%	63.97%	73.12%
- w/o SL-G	**25.55%**	**64.06%**	**73.19%**
- w/o AUG	21.18%	61.46%	70.15%
- only GEN	14.40%	61.68%	67.51%

(a) Query-based Metric

Method	EX	EFM	EVM
Few-Shot LLM	36.69%	64.43%	64.71%
RAG for LLM	47.95%	74.22%	69.80%
MultiLink (Ours)	**59.12%**	85.66%	**74.01%**
- w/o Sk-G	58.82%	85.64%	73.75%
- w/o SL-G	59.03%	**85.69%**	73.83%
- w/o AUG	55.74%	84.48%	73.36%
- only GEN	47.19%	73.33%	69.71%

(b) Execution-based Metric

Table 4: Ablation study results on MultiTEND.

Few-shot LLM and Retrieval-Augmented Generation (RAG) for LLM to compare with the ablation study results. This comparison aims to demonstrate that the high performance of MultiLink is not solely reliant on the inherent capabilities of the LLM itself, but rather stems from our designed complex and effective pipeline.

The results of the ablation experiments are shown in Table 4. MultiLink with all processes included outperforms other configurations across all metrics to varying degrees. Among them, the performance of w/o Sk-G is relatively close to that of MultiLink with all processes included, while w/o SL-G demonstrates that the contextual information provided by the SL-G module is crucial, significantly aiding the LLM in generating more accurate queries. The results of w/o AUG are the lowest, proving that our data augmentation method substantially enhances the performance of each module in the model. Overall, on the EX metric, which best reflects the model's performance in real-world scenarios, MultiLink with all processes included outperforms all other configurations. This validates the effective contribution of all components in MultiLink to the overall framework.

Furthermore, on the critical EX and EM metrics, MultiLink with all major processes included significantly outperforms Few-shot LLM, RAG for LLM, only GEN, and w/o AUG configurations. This indicates that the high accuracy of MultiLink does not directly stem from the inherent understanding and generation capabilities of the LLM itself, but rather primarily from the framework itself and the information provided by the SLM enhanced using our designed data augmentation method.

7 Conclusion

In this work, we introduce MultiTEND, a large-scale multilingual benchmark dataset for Text-to-NoSQL tasks encompassing six languages. To create this dataset, we developed a robust process that combines the capabilities of LLMs with human efforts. This approach ensures high-quality, semantically aligned, and contextually accurate database fields, NLQs, and NoSQL queries through thorough manual verification. Next, we identify the inherent challenges of multilingual Text-to-NoSQL tasks, including lexical variations and structural inconsistencies across languages. To address these issues, we propose MultiLink, a unified multilingual pipeline that breaks down the complex task into manageable steps, such as multilingual query augmentation and language-specific schema linking. Extensive experiments demonstrate that MultiLink excels in generating accurate and semantically consistent NoSQL queries across multiple languages, significantly outperforming existing baseline models.

Building on this line of research, we aim to explore additional methodologies for text-to-NoSQL tasks as the next phase of our work. We anticipate that this work will not only contribute to the ongoing evolution of the NoSQL field but also inspire further innovations, fostering a dynamic research landscape similar to the advancements seen in the parallel text-to-SQL domain.

8 Limitation

We propose a unified multilingual Text-to-NoSQL pipeline, which effectively addresses the lexical and structural challenges in multilingual NoSQL generation by integrating context information generated from fine-tuned SLMs and adopting a multi-step approach that combines CoT and RAG prompting methods. Additionally, our designed data augmentation method further enhances the accuracy and quality of NoSQL query generation by the framework. However, our research in multilingual aspects is still limited to six languages (English, German, French, Russian, Japanese, and Mandarin Chinese), which only cover a portion of the mainstream languages within the Indo-European and Sino-Tibetan language families, while neglecting the needs of other language families. The experimental results are constrained by the limited scope of general-purpose LLMs. For instance, although we use relatively advanced and high-performance LLMs in our experiments, there is a lack of exploration into methods that could enable lower-performance but more cost-efficient LLMs to achieve similar results on this task. The pipeline requires high computational costs for LLMs. For example, in scenarios where LLMs are used in the pipeline, for obtaining higher-quality outputs, the long-context inputs with rich examples and step-by-step reasoning outputs, significantly increases token overhead. Therefore, future research could expand to include more widely used languages, explore the application of Text-to-NoSQL in low-resource or minority languages, and investigate the use of other LLM architectures or the development of more cost-effective and high-performance neural-based framework strategies.

References

Divyakant Agrawal, Amr El Abbadi, Sudipto Das, and Aaron J Elmore. 2011. Database scalability, elasticity, and autonomy in the cloud. In *International conference on database systems for advanced applications*, pages 2–15. Springer.

Aseem Arora, Shabbirhussain Bhaisaheb, Harshit Nigam, Manasi Patwardhan, Lovekesh Vig, and Gautam Shroff. 2023. Adapt and decompose: Efficient generalization of text-to-SQL via domain adapted least-to-most prompting. In *Proceedings of the 1st GenBench Workshop on (Benchmarking) Generalisation in NLP*, pages 25–47, Singapore. Association for Computational Linguistics.

Christopher Baik, Zhongjun Jin, Michael Cafarella, and HV Jagadish. 2020. Duoquest: A dual-specification system for expressive sql queries. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 2319–2329.

Jagdev Bhogal and Imran Choksi. 2015. Handling big data using nosql. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, pages 393–398.

Ruichu Cai, Boyan Xu, Zhenjie Zhang, Xiaoyan Yang, Zijian Li, and Zhihao Liang. 2018. An encoder-decoder framework translating natural language to database queries. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 3977–3983. AAAI Press.

Rick Cattell. 2011. Scalable sql and nosql data stores. *Acm Sigmod Record*, 39(4):12–27.

Miguel Diogo, Bruno Cabral, and Jorge Bernardino. 2019. Consistency models of nosql databases. *Future Internet*, 11(2):43.

Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, Jinshu Lin, Dongfang Lou, et al. 2023. C3: Zero-shot text-to-sql with chatgpt. *arXiv preprint arXiv:2307.07306*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John Drake, and Qiaofu Zhang. 2021. Natural SQL: Making SQL easier to infer from natural language specifications. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2030–2042, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-sql empowered by large language models: A benchmark evaluation. *Proc. VLDB Endow.*, 17(5):1132–1145.

Jing Han, Haihong E, Guan Le, and Jian Du. 2011. Survey on nosql database. In *2011 6th International Conference on Pervasive Computing and Applications*, pages 363–366.

Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Yanyang Li, Bowen Li, Jian Sun, and Yongbin Li. 2022. S^2SQL: Injecting syntax to question-schema interaction graph encoder for text-to-SQL parsers. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1254–1262, Dublin, Ireland. Association for Computational Linguistics.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. KaggleDBQA: Realistic evaluation of text-to-SQL parsers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2261–2273, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Fei Li and Hosagrahar V Jagadish. 2014a. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84.

Fei Li and Hosagrahar V Jagadish. 2014b. Nalir: an interactive natural language interface for querying relational databases. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 709–712.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13067–13075.

Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. 2023b. Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13076–13084.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C.C. Chang, Fei Huang, Reynold

Cheng, and Yongbin Li. 2023c. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4870–4888, Online. Association for Computational Linguistics.

Hu Liu, Yuliang Shi, Jianlin Zhang, Xinjun Wang, Hui Li, and Fanyu Kong. 2023. Multi-hop relational graph attention network for text-to-sql parsing. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Xiping Liu and Zhao Tan. 2023. Divide and prompt: Chain of thought prompting for text-to-sql. *arXiv preprint arXiv:2304.11556*.

Jinwei Lu, Yuanfeng Song, Zhiqian Qin, Haodi Zhang, Chen Zhang, and Raymond Chi-Wing Wong. 2025. Bridging the gap: Enabling natural language queries for nosql databases through text-to-nosql translation.

Yuyu Luo, Nan Tang, Guoliang Li, Chengliang Chai, Wenbo Li, and Xuedi Qin. 2021a. Synthesizing natural language to visualization (nl2vis) benchmarks from nl2sql benchmarks. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, page 1235–1247, New York, NY, USA. Association for Computing Machinery.

Yuyu Luo, Nan Tang, Guoliang Li, Chengliang Chai, Wenbo Li, and Xuedi Qin. 2021b. Synthesizing natural language to visualization (nl2vis) benchmarks from nl2sql benchmarks. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1235–1247.

ABM Moniruzzaman and Syed Akhter Hossain. 2013. Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. *arXiv preprint arXiv:1307.0191*.

Lakshmi Nivas Nalla and Vijay Mallik Reddy. 2022. Sql vs. nosql: Choosing the right database for your ecommerce platform. *International Journal of Advanced Engineering Technologies and Innovations*, 1(2):54–69.

Octavian Popescu, Irene Manotas, Ngoc Phuoc An Vo, Hangu Yeo, Elahe Khorashani, and Vadim Sheinin. 2022. Addressing limitations of encoder-decoder based approach to text-to-sql. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1593–1603.

Mohammadreza Pourreza and Davood Rafiei. 2024. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems*, 36.

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. RASAT: Integrating relational structures into pretrained Seq2Seq model for text-to-SQL. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3215–3229, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Abdul Quamar, Vasilis Efthymiou, Chuan Lei, Fatma Özcan, et al. 2022. Natural language interfaces to data. *Foundations and Trends® in Databases*, 11(4):319–414.

Ohad Rubin and Jonathan Berant. 2021. SmBoP: Semi-autoregressive bottom-up semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 311–324, Online. Association for Computational Linguistics.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jaydeep Sen, Chuan Lei, Abdul Quamar, Fatma Özcan, Vasilis Efthymiou, Ayushi Dalmia, Greg Stager, Ashish Mittal, Diptikalyan Saha, and Karthik Sankaranarayanan. 2020. Athena++ natural language querying for complex nested sql queries. *Proceedings of the VLDB Endowment*, 13(12):2747–2759.

Michael Staniek, Raphael Schumann, Maike Züfle, and Stefan Riezler. 2024. Text-to-OverpassQL: A natural language interface for complex geodata querying of OpenStreetMap. *Transactions of the Association for Computational Linguistics*, 12:562–575.

Van-Dai Ta, Chuan-Ming Liu, and Goodwill Wandile Nkabinde. 2016. Big data stream computing in healthcare real-time analytics. In *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pages 37–42.

Chang-Yu Tai, Ziru Chen, Tianshu Zhang, Xiang Deng, and Huan Sun. 2023. Exploring chain of thought style prompting for text-to-SQL. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5376–5393, Singapore. Association for Computational Linguistics.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL:

Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, LinZheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. 2025. MAC-SQL: A multi-agent collaborative framework for text-to-SQL. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 540–557, Abu Dhabi, UAE. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yanzheng Xiang, Qian-Wen Zhang, Xu Zhang, Zejie Liu, Yunbo Cao, and Deyu Zhou. 2023. G3r: A graph-guided generate-and-rerank framework for complex and cross-domain text-to-sql generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 338–352.

Yuanzhen Xie, Xinzhou Jin, Tao Xie, Matrixmxlin Matrixmxlin, Liang Chen, Chenyun Yu, Cheng Lei, Chengxiang Zhuo, Bo Hu, and Zang Li. 2024. Decomposition for enhancing attention: Improving LLM-based text-to-SQL through workflow paradigm. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10796–10816, Bangkok, Thailand. Association for Computational Linguistics.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. SQL-to-text generation with graph-to-sequence model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 931–936, Brussels, Belgium. Association for Computational Linguistics.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, bailin wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, richard socher, and Caiming Xiong. 2021. Gra{pp}a: Grammar-augmented pre-training for table semantic parsing. In *International Conference on Learning Representations*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Jiaan Zeng. 2015. *Resource sharing for multi-tenant nosql data store in cloud*. Ph.D. thesis, Indiana University.

Chao Zhang, Yuren Mao, Yijiang Fan, Yu Mi, Yunjun Gao, Lu Chen, Dongfang Lou, and Jinshu Lin. 2024a. Finsql: Model-agnostic llms-based text-to-sql framework for financial analysis. In *Companion of the 2024 International Conference on Management of Data*, SIGMOD/PODS '24, page 93–105, New York, NY, USA. Association for Computing Machinery.

Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. 2023. ACT-SQL: In-context learning for text-to-SQL with automatically-generated chain-of-thought. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3501–3532, Singapore. Association for Computational Linguistics.

Hanchong Zhang, Ruisheng Cao, Hongshen Xu, Lu Chen, and Kai Yu. 2024b. CoE-SQL: In-context learning for multi-turn text-to-SQL with chain-of-editions. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6487–6508, Mexico City, Mexico. Association for Computational Linguistics.

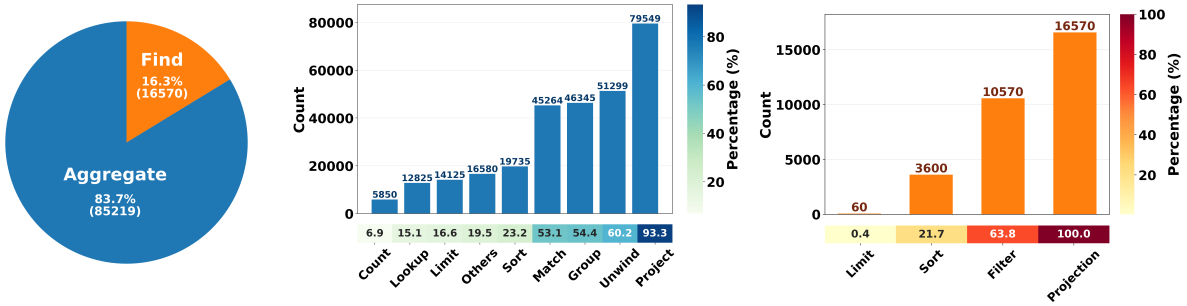
Yanzhao Zheng, Haibin Wang, Baohua Dong, Xingjun Wang, and Changshan Li. 2022. HIE-SQL: History information enhanced network for context-dependent text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2997–3007, Dublin, Ireland. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A Dataset Analysis

A.1 Dataset Details



(a) NoSQL Query Method (b) Stages in Aggregate Method (c) Operations in Find Method

Figure 4: NoSQL Query Statistics in MultiTEND

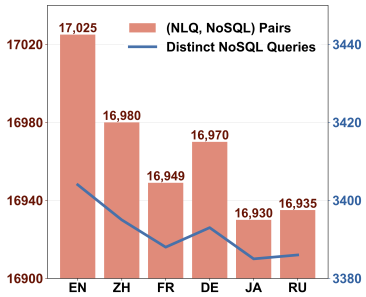


Figure 5: (NLQ,Query) Statistics

#-Database	#-Domain	#-Collections	#-Documents	#-Fields
924	105	2082	197874	35760
Top-5 Domains				
Sport	Customer	School	Shop	Student
Statistics of Database				
	#-Avg	#-Max	#-Min	#-Total
Cols	2.25	7	1	2082
Docs	214.15	13694	3	197874
Fields	38.7	331	7	35760

Table 5: Database Statistics in MultiTEND

Figure 5 presents detailed statistics of (NLQ, NoSQL) Pairs and distinct NoSQL Queries across different languages in MultiTEND. Figure 4 displays the statistics of NoSQL queries in MultiTEND (covering all six languages). Specifically, figure 4a uses a pie chart to illustrate the distribution of different query methods in MultiTEND, while figure 4b and figure 4c show the counts of stages in the aggregate method and operators in the find method, respectively, with a heatmap included to represent the proportion of queries that use each specific stage or operator relative to the total number of queries employing the corresponding method.

Table 5 conveys detailed statistical information about all databases (covering six languages) in the MultiTEND dataset, which includes a total of 924 databases spanning 105 domains. The top five most represented domains in the dataset are Sport, Customer, School, Shop, and Student. Across all databases, there are a total of 2,082 collections, 197,874 documents, and 35,760 fields. Each database contains an average of 2.25 collections, with values ranging from 1 to 7. The number of documents averages 214.15 per database, spanning from 3 to 13,694. Field counts average 38.7 per database, varying from 7 to 331.

A.2 Analysis and Findings

Metric	Model	EN	ZH	FR	DE	JA	RU	AVG (5 langs)
EM	Fine-tuned Llama	17.05%	13.57%	16.53%	15.78%	16.40%	14.51%	15.36%
	Zero-shot LLM	0.29%	0.61%	0.61%	0.54%	0.54%	0.29%	0.52%
	RAG for LLM	16.09%	13.98%	15.62%	14.33%	12.02%	13.89%	13.97%
	SMART	18.85%	13.94%	18.38%	18.30%	18.05%	15.89%	16.91%
QSM	Fine-tuned Llama	57.19%	56.71%	54.22%	56.14%	56.22%	54.91%	55.64%
	Zero-shot LLM	51.24%	47.76%	50.36%	50.43%	47.35%	48.95%	48.97%
	RAG for LLM	62.30%	59.52%	60.51%	60.17%	57.36%	58.87%	59.28%
	SMART	61.15%	57.69%	61.23%	59.35%	58.11%	57.17%	58.71%
QFC	Fine-tuned Llama	60.76%	53.83%	56.61%	62.35%	58.70%	58.41%	57.98%
	Zero-shot LLM	60.29%	58.59%	58.92%	60.07%	58.09%	60.22%	59.18%
	RAG for LLM	68.04%	67.86%	67.03%	67.47%	65.29%	67.46%	67.02%
	SMART	65.05%	60.36%	60.97%	63.86%	62.03%	59.34%	61.31%
EX	Fine-tuned Llama	44.61%	36.86%	41.26%	41.44%	43.32%	38.23%	40.22%
	Zero-shot LLM	36.58%	28.99%	33.86%	34.91%	30.63%	29.68%	31.61%
	RAG for LLM	51.70%	47.02%	49.28%	48.59%	45.12%	45.99%	47.20%
	SMART	48.86%	38.05%	44.69%	44.22%	43.30%	41.03%	42.26%
EFM	Fine-tuned Llama	84.97%	78.84%	80.14%	81.44%	79.50%	83.54%	80.69%
	Zero-shot LLM	51.78%	54.40%	57.36%	57.11%	57.01%	58.19%	56.82%
	RAG for LLM	72.76%	73.60%	73.88%	74.31%	75.02%	75.73%	74.51%
	SMART	86.74%	83.10%	85.67%	83.72%	84.76%	86.31%	84.71%
EVM	Fine-tuned Llama	74.20%	66.28%	68.38%	70.72%	68.43%	74.73%	69.71%
	Zero-shot LLM	58.05%	57.47%	59.93%	60.14%	60.14%	59.60%	59.46%
	RAG for LLM	70.38%	68.47%	70.40%	70.41%	68.14%	70.98%	69.68%
	SMART	76.79%	75.85%	78.99%	73.68%	74.60%	78.30%	76.28%

Table 6: Comparison of baseline model's performance across multiple languages on **MultiTEND** dataset based on the metrics. Notice that **AVG** is the average value of the corresponding metric across the 5 **non-English** languages

Dataset	Model	EM	QSM	QFC	EX	EFM	EVM
TEND	SMART	23.82%	63.21%	75.60%	65.08%	87.21%	72.79%

Table 7: Performance of SMART on **TEND** dataset based on the metrics.

To clarify the challenges posed by multilingual Text-to-NoSQL tasks to existing models, we first fine-tuned the Llama-3.2-1B model for testing and found its accuracy to be very low. Through detailed experiments (as shown in Figure 6), we further analyzed the additional challenges multilingual Text-to-NoSQL tasks pose to existing models. Results show that the common errors in NoSQL query generation from different models are primarily caused or worsened by multilingual issues. Additionally, we compared the performance of SMART (Lu et al., 2025), a model specifically designed for Text-to-NoSQL, on a pure English dataset (TEND) and a mixed dataset of six languages (MultiTEND)(See Table refappx.tab:Performence Comparison of baselines on MultiTEND and Table 7). Results show that despite being designed for NoSQL generation, the quality of NoSQL queries generated by SMART significantly drops in multilingual contexts. Based on the experimental results, we analyzed the factors leading to the decline in NoSQL generation quality in multilingual environments, with the findings as follows:

- **Finding 1**: In the process of identifying entity mentions in NLQ and mapping them to corresponding database fields (i.e., schema linking), significant lexical differences across languages, especially in those with more complex lexical formation rules (such as Japanese hiragana and katakana, Russian Cyrillic characters, and the morphological variations in German and French), impose higher demands on the model's language understanding capabilities, resulting in a significant drop in mapping accuracy.
- **Finding 2**: The support for nested documents and array structures in NoSQL requires models to have a stronger understanding of database schemas in multilingual environments to handle complex nested fields or situations requiring array expansion. This makes the already challenging schema linking task even more difficult due to

the multilingual context.

- **Finding 3**: In multilingual contexts, NLQs often exhibit vastly different syntactic structures due to language differences, significantly increasing the difficulty for models to comprehend multilingual questions. This also leads to errors in two critical tasks: mapping intentions to operators (intention mapping task) and mapping intentions to database fields (schema linking task).

Based on these experiments and findings, we categorize the challenges in MultiTEND into Structural Challenge and Lexical Challenge.

B More Implementation Details

Dataset Construction During the dataset construction process, we utilized the "gpt-4o-mini-2024-07-18" model to extend the dataset from English to multiple languages, with the parameter setting temperature = 0.0.

MultiLink When building the RAG vector library, we construct a corresponding vector library for the training set of each language. In the vector library construction process, the text-to-embedding model used is "text-embedding-ada-002", and the Faiss library is employed to build indexes for efficient vector search.

In the data augmentation phase of MultiLink, we use the "DeepSeek-V3" model to expand multilingual data pairs, with the parameter setting 'temperature = 0.0'.

In the Parallel Multilingual Sketch-Schema Prediction phase of MultiLink, all SLMs are fine-tuned based on "Llama-3.2-1B" using a full-parameter fine-tuning strategy with AdamW as the optimizer. We set the hyperparameters for fine-tuning as follows: batch size = 2, learning rate = 5e-5, and epochs = 3. Additionally, we configure the gradient accumulation steps to 8 and set the maximum input token length to 2048. For the generation of SLMs, we use top-p = 0.7 and temperature = 0.0, with a maximum input token length of 2048 and a maximum output token length of 512.

In the Retrieval-Augmented Chain-of-Thought Query Generation phase of MultiLink, when retrieving examples from the vector library, we set the similarity threshold to 0.5 and the retrieval count (rag retrieve num) to 6.0. For query generation, the LLM used is "DeepSeek-V3", with the parameter setting 'temperature = 0.0'.

Baselines For baseline methods of Zero-shot LLM, Few-shot LLM, RAG for LLM, the LLM we use is "DeepSeek-V3" , with the parameter setting 'temperature = 0.0'; Specifically for RAG for LLM, we employ the same vector library as MultiLink, setting the similarity threshold to 0.5 and the retrieval count (rag retrieve num) to 6.0.

For the part of SMART that requires the use of LLM, we also use "DeepSeek-V3", with the parameter setting' temperature = 0.0', which is consistent with the configuration of MultiLink. For the part of SMART that involves fine-tuning SLM as well as the baseline method of Fine-tuned SLM, we maintain the same settings as those used for fine-tuning SLM in MultiLink, using "Llama-3.2-1B". The hyperparameters for fine-tuning are set as follows: batch size = 2, learning rate = 5e-5, and epochs = 3. Additionally, we configure the gradient accumulation steps to 8 and set the maximum input token length to 2048. For the generation of SLMs, we use top-p = 0.7 and temperature = 0.0, with a maximum input token length of 2048 and a maximum output token length of 512.

C More Experimental Details

C.1 Baselines

We utilized a variety of popular neural network models, LLM-based prompting methods,SLM-based fine-tuning methods and existing Text-to-NoSQL pipelines as baseline models for a comprehensive performance comparison with MultiLink. The baseline models are as follows:

- **Zero-shot LLM**: The zero-shot prompting approach utilizes the inherent zero-shot learning capabilities of LLM, allowing LLM to produce precise and contextually appropriate responses without the need for prior training or example-driven instructions.
- **Few-shot LLM**: The few-shot prompting technique serves as a key mechanism for in-context learning (ICL), where a limited set of examples is incorporated into the context to instruct LLM on executing tasks within specialized domains.
- **RAG for LLM**: Retrieval-Augmented Generation (RAG) technology provides an alternative approach to support LLM in downstream tasks. Unlike direct few-shot prompting, RAG dynamically retrieves relevant examples from a knowledge base based on the model's input, enriching the context and effectively reducing hallucinations induced by insufficient or ambiguous information.
- **Fine-tuned SLM**: Fine-tuning is another effective strategy for enhancing the performance of language models in specific downstream tasks, such as predicting NoSQL query generation. We fine tune SLM based on two different approaches (Monolingual Training and Multilingual Training) to compare the quality of NoSQL queries predicted by SLM based on single-target language training data and training data from multiple languages.
- **SMART**: SMART is the first and currently the only framework in the Text-to-NoSQL domain tackling the task of converting English NLQs to NoSQL queries. With the assistance of SLM and RAG technologies, it constructs four main processes: SLM-based schema prediction, SLM-based query generation, query refinement based on predicted schema and retrieved examples, and execution results-based query optimization.
- **MultiLink**: MultiLink is the framework proposed in this work, aiming to address the challenges of multilingual Text-to-NoSQL tasks. It constructs three main processes: Intention-aware Multilingual Data Augmentation (MIND), Parallel Multilingual Sketch-Schema Prediction,and Retrieval-Augmented Chain-of-Thought Query Prediction.

C.2 Evaluation Metrics

We report results using the same metrics as SMART, which include Exact Match (EM) and Execution Accuracy (EX), each with more detailed subdivisions such as Query Stages Match (QSM) and Query Fields Coverage (QFC) under EM, and Execution Fields Match (EFM) and Execution Value Match (EVM) under EX.

Here are detailed descriptions of each metric:

- **Exact Match (EM)**: The purpose of this metric is to evaluates whether the generated query is an exact match to the gold query, considering both its structure and content. It is calculated as:

$$EM = \frac{N_{em}}{N}$$

  Where N_{em} represents the count of queries fully matching the gold query, and N signifies the total number of queries within the test set. EM serves as a stringent measure of syntactic and semantic alignment.

  - **Query Stages Match (QSM)**: QSM is designed to check if the generated query's key stages (e.g., match,group, lookup) mirror the gold query in the order and keywords employed. It's calculated as:

$$QSM = \frac{N_{qsm}}{N}$$

    where N_{qsm} represents the count of queries with matching stages.
  - **Query Fields Coverage (QFC)**: QFC assesses if the generated query encompasses all the fields present in the gold query, considering both database fields and query-defined fields. It's defined as:

$$QFC = \frac{N_{qfc}}{N}$$

    Where N_{qfc} represents the number of queries with complete field coverage.
- **Execution Accuracy (EX)**: This metric evaluates the accuracy of the execution results for the generated query on the database. It is calculated as follows:

$$EX = \frac{N_{ex}}{N}$$

  where N_{ex} represents the number of queries whose execution results align with those of the gold query. EX serves as the most critical performance metric for evaluating Text-to-NoSQL models.

- **Execution Fields Match (EFM)**: EFM validates the alignment of field names derived from the execution of the generated query against those obtained from the gold query. It's defined as:

$$EFM = \frac{N_{efm}}{N}$$

  where N_{efm} represents the number of queries with matching field names in the results.
- **Execution Value Match (EVM)**: EVM evaluates the correspondence between the values in the execution results of the generated query and those in the gold query. It is defined as:

$$EVM = \frac{N_{evm}}{N}$$

  where N_{evm} represents the number of queries with matching values in the results.

D More Experimental Results

D.1 Performance Comparison

Metric	Model	EN	ZH	FR	DE	JA	RU	AVG (5 langs)
				Query-based Metric Results				
EM	Fine-tuned Llama	17.05%	13.57%	16.53%	15.78%	16.40%	14.51%	15.36%
	Zero-shot LLM	0.29%	0.61%	0.61%	0.54%	0.54%	0.29%	0.52%
	Few-shot LLM	12.18%	10.25%	10.65%	10.65%	9.87%	11.34%	10.55%
	RAG for LLM	16.09%	13.98%	15.62%	14.33%	12.02%	13.89%	13.97%
	SMART	18.85%	13.94%	18.38%	18.30%	18.05%	15.89%	16.91%
	MultiLink (Ours)	30.05%	23.47%	25.58%	25.65%	23.53%	24.95%	24.64%
QSM	Fine-tuned Llama	57.19%	56.71%	54.22%	56.14%	56.22%	54.91%	55.64%
	Zero-shot LLM	51.24%	47.76%	50.36%	50.43%	47.35%	48.95%	48.97%
	Few-shot LLM	57.01%	58.52%	56.06%	53.83%	56.14%	55.45%	56.00%
	RAG for LLM	62.30%	59.52%	60.51%	60.17%	57.36%	58.87%	59.28%
	SMART	61.15%	57.69%	61.23%	59.35%	58.11%	57.17%	58.71%
	MultiLink (Ours)	65.91%	62.19%	64.71%	64.73%	63.26%	63.29%	63.63%
QFC	Fine-tuned Llama	60.76%	53.83%	56.61%	62.35%	58.70%	58.41%	57.98%
	Zero-shot LLM	60.29%	58.59%	58.92%	60.07%	58.09%	60.22%	59.18%
	Few-shot LLM	62.88%	63.32%	59.39%	58.38%	61.51%	61.23%	60.76%
	RAG for LLM	68.04%	67.86%	67.03%	67.47%	65.29%	67.46%	67.02%
	SMART	65.05%	60.36%	60.97%	63.86%	62.03%	59.34%	61.31%
	MultiLink (Ours)	76.55%	71.14%	73.22%	73.73%	72.18%	72.22%	72.50%
				Execution-based Metric Results				
EX	Fine-tuned Llama	44.61%	36.86%	41.26%	41.44%	43.32%	38.23%	40.22%
	Zero-shot LLM	36.58%	28.99%	33.86%	34.91%	30.63%	29.68%	31.61%
	Few-shot LLM	40.79%	34.95%	36.64%	37.08%	35.93%	34.77%	35.87%
	RAG for LLM	51.70%	47.02%	49.28%	48.59%	45.12%	45.99%	47.20%
	SMART	48.86%	38.05%	44.69%	44.22%	43.30%	41.03%	42.26%
	MultiLink (Ours)	67.64%	57.71%	59.86%	58.90%	55.75%	54.88%	57.42%
EFM	Fine-tuned Llama	84.97%	78.84%	80.14%	81.44%	79.50%	83.54%	80.69%
	Zero-shot LLM	51.78%	54.40%	57.36%	57.11%	57.01%	58.19%	56.82%
	Few-shot LLM	63.21%	63.79%	62.85%	65.31%	64.47%	66.97%	64.68%
	RAG for LLM	72.76%	73.60%	73.88%	74.31%	75.02%	75.73%	74.51%
	SMART	86.74%	83.10%	85.67%	83.72%	84.76%	86.31%	84.71%
	MultiLink (Ours)	88.92%	85.41%	84.64%	85.27%	85.32%	84.40%	85.01%
EVM	Fine-tuned Llama	74.20%	66.28%	68.38%	70.72%	68.43%	74.73%	69.71%
	Zero-shot LLM	58.05%	57.47%	59.93%	60.14%	60.14%	59.60%	59.46%
	Few-shot LLM	65.37%	63.21%	63.94%	66.35%	63.46%	65.96%	64.58%
	RAG for LLM	70.38%	68.47%	70.40%	70.41%	68.14%	70.98%	69.68%
	SMART	76.79%	75.85%	78.99%	73.68%	74.60%	78.30%	76.28%
	MultiLink (Ours)	76.33%	74.32%	74.42%	73.55%	72.03%	73.38%	73.54%

Table 8: Comparison of each model's performance across multiple languages on MultiTEND based on the metrics. Notice that **AVG** is the average value of the corresponding metric across the 5 **non-English** languages.

As shown in Table 8, models consistently achieve higher performance in English across all metrics, suggesting baseline models are more proficient in handling complex tasks in English. This can be attributed to their extensive training on English data, giving them an inherent advantage over other languages. In contrast, performance in Chinese and Russian often lags behind languages like English, Japanese, French, and German. This disparity may arise not only from limited training data for these languages but also from the unique challenges posed by Chinese character construction, syntax, and the Cyrillic alphabet in Russian, which introduce additional complexities in comprehension and generation.

According to the results shown in Table 8 and Table 9, there are significant differences in the performance of

Model	EM	QSM	QFC	EX	EFM	EVM
Fine-tuned Llama	15.64%	55.90%	58.44%	40.95%	81.41%	70.46%
Zero-shot LLM	0.48%	49.35%	59.36%	32.44%	55.98%	59.22%
Few-shot LLM	10.82%	56.17%	61.12%	36.69%	64.43%	64.71%
RAG for LLM	14.32%	59.79%	67.19%	47.95%	74.22%	69.80%
SMART	17.23%	59.12%	61.94%	43.36%	85.05%	**76.37%**
MultiLink (Ours)	**25.54%**	**64.01%**	**73.17%**	**59.12%**	**85.66%**	74.01%

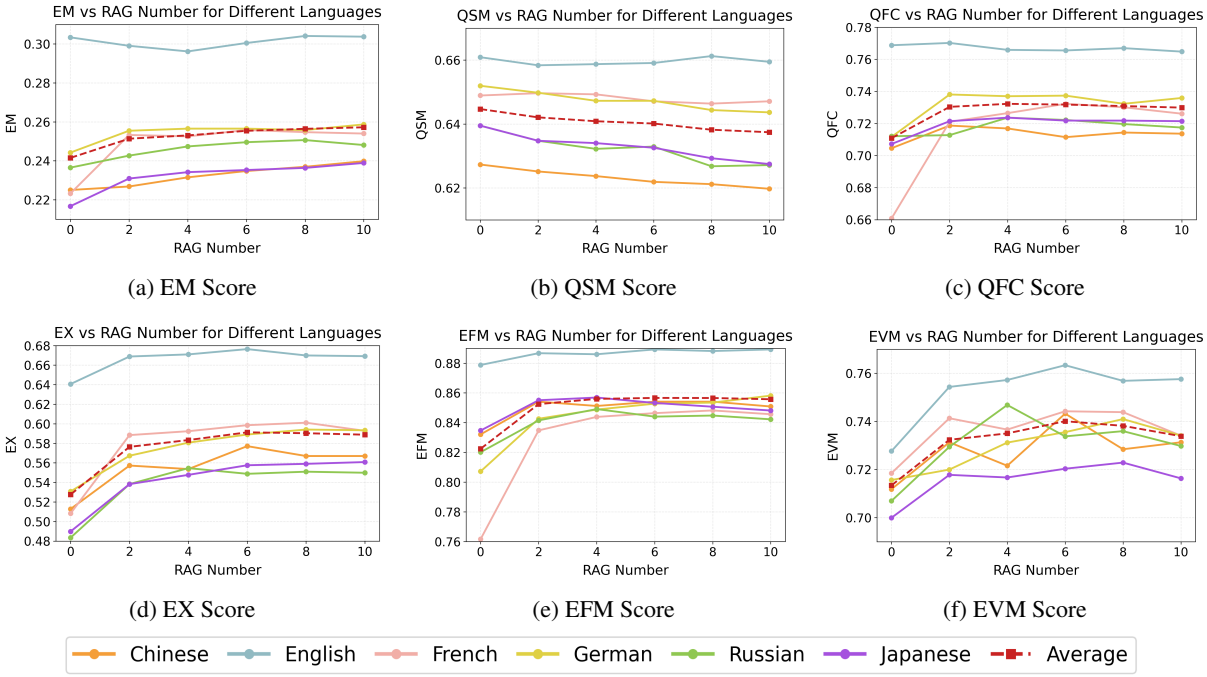
Table 9: Comparison of each model's average performance across six languages(**AVG of 6 langs**) on MultiTEND based on the metrics.

various models on key metrics. Our model performs exceptionally well across all languages and metrics except for EVM, outperforming the second-best baseline model by approximately 5% to 20%. Particularly on the two key metrics for real-world applications, EM and EX, our model maintains an absolute lead with average advantages of 25.54% and 59.12%, respectively, across all languages. In contrast, the Zero-shot LLM method performs the worst across all languages and metrics, particularly on EM and EX, with its average accuracy across six languages trailing MultiLink by nearly 25%. The other three methods achieve average accuracies across the six languages on EM of 15.64% (Fine-tuned), 14.32% (RAG for LLM), and 10.82% (Zero-shot LLM), while on EX, their average accuracies are 47.95% (RAG for LLM), 40.95% (Fine-tuned Llama), and 36.69% (Few-shot LLM). Overall, our model demonstrates significant advantages across all metrics on every language.

Further analysis of Table 9 reveals distinct performance differences between model fine-tuning and direct prompting LLM methods on the MultiTEND test set under cross-domain criteria. Fine-tuned Llama achieved an 81.41% accuracy in EFM, outperforming RAG for LLM by 7.19%, indicating its stronger capability in generating queries that retrieve correct field results. On the other hand, RAG for LLM surpassed Fine-tuned Llama by margins ranging from 3.8% to 8.75% in EX, QSM, and QFC metrics, demonstrating its superior understanding of the mapping relationships between NLQs and NoSQL database fields, as well as a deeper grasp of data operations in NoSQL queries, leading to higher execution accuracy

Among the Multi-Step methods specifically designed for Text-to-NoSQL tasks, the performance gap between SMART and our proposed MultiLink approach is significant. As shown in Table 8, specifically, in terms of performance across every language, MultiLink outperforms SMART, which is tailored for monolingual NoSQL generation, across all metrics except EVM, with margins ranging from 2.18% to 18.78%. This demonstrates that SMART, designed exclusively for English contexts, cannot directly address the challenges posed by multilingual tasks. In contrast, the intention mapping and schema linking methods designed in MultiLink, specifically developed to tackle multilingual generation challenges, effectively overcome these difficulties, enabling it to achieve significantly better performance in multilingual scenarios compared to other baseline models.

D.2 Parameter Study



(a) EM Score (b) QSM Score (c) QFC Score

(d) EX Score (e) EFM Score (f) EVM Score

Figure 6: Parameter study of RAG number on the MultiTEND test set.

Figure 6 illustrates the performance variations of MultiLink across multiple languages under different numbers of retrieved examples for all metrics. The vertical axis represents the model's accuracy under each metric, and the horizontal axis represents the RAG number ranging from 0 to 10.

Analyzing Figure 6 and observing the changes in the average represented by the red dashed line, we find that as the RAG num increases, MultiLink shows significant improvement on all metrics except QSM, followed by a slow rise and a gradual decline after reaching a certain value. The initial improvement, resulting from the change in RAG num from 0 to 2, indicates that RAG greatly enhances model performance. The subsequent decline in MultiLink's performance beyond a certain value might be attributed to excessive context length introducing redundant information, which interferes with generation.

Moreover, the model's performance on the QSM metric declines as the RAG num increases, which may be attributed to the influence of retrieved examples on the model's decisions regarding NoSQL operations. However, considering the performance on other metrics, the retrieved examples contribute to the model generating more accurate queries before the RAG num reaches 6, with the peak accuracy of model execution occurring at a RAG num of 6.

D.3 Case Study

Table 10 presents a case study comparing various baseline methods and MultiLink in generating NoSQL queries and their execution results. Firstly, we observe that the fine-tuned Llama exhibits a significant lack of understanding regarding database field names and overall structure, as illustrated in Table 10. The model not only fails to accurately match target fields (such as "课程名称"), leading to erroneous query logic and invalid results, but also incorrectly identifies "课程" as an independent collection rather than recognizing it as a field nested within the "科目" collection. Additionally, it lacks the ability to deconstruct nested fields (e.g., using $unwind), rendering it incapable of properly handling array-type nested structures.

On the other hand, RAG for LLM performs well in handling the syntax of complex operations, but it still falls short in understanding field names and matching database schemas, and it fails to accurately incorporate the actual values in the database for retrieval. As shown in Table 10, although the model correctly uses `$unwind` to deconstruct nested fields, the query still cannot execute correctly because it does not filter based on the actual values in the database.

In comparison, SMART demonstrates a solid grasp of NoSQL syntax and shows a slightly better understanding of user query intentions and database structures compared to methods that directly utilize LLMs through prompting for generation. However, it struggles with handling more complex fields that involve nested relationships. Due to a lack of multilingual knowledge, it often incorrectly maps NLQ to database fields, finding it difficult to distinguish between semantically similar fields in different languages. As illustrated in Table 10, SMART erroneously selected the "课程" collection instead of the "科目" collection and failed to correctly structure the nested fields. In contrast, MultiLink fully and accurately comprehended the query intent even in multilingual contexts, with clear query logic and the ability to generate queries that precisely align with the query intent. This indicates that MultiLink can effectively understand and execute multilingual Text-to-NoSQL tasks and generalizes well to test set examples after learning from the training set.

NLQ: 学生在西班牙语课程的注册日期是什么时候? **(When is the registration date for students in the Spanish course?)**		
Target NoSQL	db.科目.aggregate([{ $unwind: "$课程" }, { $match: { "课程.课程名称": "Spanish" } }, { $unwind: "$课程.学生课程注册" }, { $project: { 注册的日期: "$课程.学生课程注册.注册的日期", _id: 0 } }]);	Standard query: Correct usage of nested field paths Results: [{'注册的日期' : '2017-12-07 02:21:13'},...]
Fine-tuned Llama	db.课程.aggregate([{ $match: { 课程名称: "Spanish" } }, { $unwind: "$学生课程注册" }, { $project: { 注册的日期: "$学生课程注册.注册的日期", _id: 0 } }]);	Error: Incorrect collection ('科目' vs. '课程') Results: []
RAG for LLM	db.科目.aggregate([{ $unwind: "$课程" }, { $match: { "课程.课程名称": "西班牙语" } }, { $unwind: "$课程.学生课程注册" }, { $project: { 注册日期: "$课程.学生课程注册.注册的日期", _id: 0 } }]);	Error: Wrong value ('Spanish' vs. '西班牙语') Results: []
SMART	db.课程.aggregate([{ $match: { 课程名称: "Spanish" } }, { $unwind: "$学生课程注册" }, { $project: { 注册的日期: "$学生课程注册.注册的日期", _id: 0 } }]);	Error: Wrong collection and field paths Results: []
MultiLink (Ours)	db.科目.aggregate([{ $unwind: "$课程" }, { $match: { "课程.课程名称": "Spanish" } }, { $unwind: "$课程.学生课程注册" }, { $project: { 注册的日期: "$课程.学生课程注册.注册的日期", _id: 0 } }]);	Success: Correct structure and values Results: [{'注册的日期' : '2017-12-07 02:21:13'},...]

Table 10: **Case Study: Comparison of Different Approaches in Complex Nested Queries.** This table illustrates the performance of three baseline methods against our MultiLink method in generating Chinese MongoDB queries based on the same NLQ.

E Prompt Examples

In this section, we present the specific prompts designed for each LLM application scenario within MultiLink.

E.1 Prompt Design in Data Construction Pipeline

E.1.1 DB Fields Translation in Dataset Construction

Prompt for DB Fields Translation in Dataset Construction

Role: SYSTEM
Content: You are a language translator,just translate every English key name in the input dictionary to target language and no more talk, your translation should be as accurate as possible, do not make it up.

Role: USER
Content:
Given the input database schema and key list, perform the following actions:
1. Based on the understanding of the database structure below, translate the key names in the key list from English to targert language,and return them in a list.
2. Ensure that identical English key names within the list are consistently translated into the same target language keys.
3. When you encounter a key name with an underscore, first understand the semantic meaning of the key before translating it.
   For example, when the target language is Chinese:
   original key : investor_id -> semantic meaning : id of inverstor -> translated key : 投资者_id
   original key : date_of_transaction -> semantic meaning : date of transaction -> translated key : 交易日期

   For example, when the target language is German:
   original key : investor_id -> semantic meaning : id of inverstor -> translated key : Anleger_id
   original key : date_of_transaction -> semantic meaning : date of transaction -> translated key : Geschäftsdatum
4. Ensure that every key name in the key list is translated and organized into the output list.The final output format should be like:
```json
[translated_key1,translated_key2,...]
```
Target language:
Chinese
Input Database Schema:
{"Activity": {"actid": "INTEGER", "activity_name": "varchar(25)", "Participates_in": [{"stuid": "INTEGER", "actid": "INTEGER"}], "Faculty_Participates_in": [{"FacID": "INTEGER", "actid": "INTEGER"}]}, "Student": {"StuID": "INTEGER", "LName": "VARCHAR(12)", "Fname": "VARCHAR(12)", "Age": "INTEGER", "Sex": "VARCHAR(1)", "Major": "INTEGER", "Advisor": "INTEGER", "city_code": "VARCHAR(3)", "Participates_in": [{"stuid": "INTEGER", "actid": "INTEGER"}]}, "Faculty": {"FacID": "INTEGER", "Lname": "VARCHAR(15)", "Fname": "VARCHAR(15)", "Rank": "VARCHAR(15)", "Sex": "VARCHAR(1)", "Phone": "INTEGER", "Room": "VARCHAR(5)", "Building": "VARCHAR(13)", "Faculty_Participates_in": [{"FacID": "INTEGER", "actid": "INTEGER"}]}}
Key List:
["Faculty_Participates_in", "actid", "LName", "Faculty", "Student", "city_code", "Fname", "stuid", "Participates_in", "Sex", "activity_name", "Major", "Activity", "Age", "FacID", "Phone", "Building", "StuID", "Rank", "Room", "Lname", "Advisor"]

E.1.2 NLQ Translation in Dataset Construction

Prompt for NLQ Translation in Dataset Construction

Role: USER
Content:
#Given the Original Natrual Language Query List and Corresponding MongoDB Query, perform the following actions: ...
A: Let's think step by step!

Role: ASSISTANT
Content:
Step 1: Identify Words for Translation ...
Final Translations ...

Role: USER
Content:
#Given the Original Natrual Language Query List and Corresponding MongoDB Query, perform the following actions:
1. Recognize the words that should be translated into Chinese.
2. Analyze the NLQ with the corresponding MongoDB query,Find the value of the MongoDB Query index in the natural sentence, maintain the consistency of this value's expression with that in the MongoDB Query.
3. Ensure the translation flows smoothly and naturally, with consistency in vocabulary.
4. Based on the steps above, Translate every NLQ in the List into the Target Language.
5. Output the translated NLQ in the following format:
```markdown
1. **Ori_NLQ1**: [Translated_NLQ1],
2. **Ori_NLQ2**: [Translated_NLQ2],
...
```
Original Natrual Language Query List:
```markdown
1. **Ori_NLQ1**: Show the booking status code and the corresponding number of bookings.
2. **Ori_NLQ2**: How many bookings does each booking status have? List the booking status code and the number of corresponding bookings.
3. **Ori_NLQ3**: Display the booking status codes along with the total number of bookings for each status.
4. **Ori_NLQ4**: What is the count of bookings for each booking status? Provide the booking status codes and their respective counts.
5. **Ori_NLQ5**: List the different booking status codes and the associated number of bookings for each status.
```
Corresponding MongoDB Query:
```javascript
db.Apartment_Buildings.aggregate([{$unwind:"$Apartments"},{$unwind:"$Apartments.Apartment_Bookings"},{$group:{_id:"$Apartments.Apartment_Bookings.booking_status_code",COUNT:{$sum:1}}},{$project:{booking_status_code:"$_id",COUNT:1,_id:0}}]);
```
A: Let's think step by step!

E.2 Prompt Design in MultiLink

E.2.1 Intention-aware Multilingual Data Augmentation (MIND)

Prompt for Data Pairs Synthesis in Data Augmentation

Role: SYSTEM
Content: You are an excellent MongoDB expert responsible for writing corresponding MongoDB queries based on Natural Language Query (NLQ) and the Database Schema.

Role: USER
Content:
Given the MongoDB Schema, MongoDB Data, Original MongoDB Query and its corresponding Natural Language Queries (NLQs), Write another MongoDB Query (MQL) and its NLQs that have different intent from the original MongoDB Query by performing the following steps:
1. Analyze the structural relationships between collections and fields within the MongoDB Schema.
2. Analyze the fields in the MongoDB collection referenced by the existing NLQ, based on the MongoDB schema, NLQ, and schema linking results, to identify the logical relationships between these fields and collections that support the NLQ's underlying query logic.
3. Analyze the correspondence between the querying intent of the Original MongoDB Query (MQL) and Original NLQs.
4. Based on the above analysis results and MongoDB Data, write a new MQL with a completely different querying intent from the Original MongoDB Query (MQL).
5. Write one NLQ corresponding to the generated MQL, and expand 4 more new NLQs that have the same meaning but different expressions from the generated NLQ.
6. Return the new MQL and its corresponding NLQs in the following format:
- Generated MongoDB Query (MQL):
```javascript
<MongoDB Query>
```
- Corresponding Natural Language Queries (NLQs):
```markdown
NLQ1: <Natural Language Query 1>
NLQ2: <Natural Language Query 2>
...
NLQ5: <Natural Language Query 5>
```
Original MongoDB Query
...
Original Natural Language Queries (NLQs)
['Find the students who have failed tests in any course, along with the course and subject details.', 'Retrieve the list of students who failed tests in specific courses, including the course and subject information.', ...]
MongoDB Schema
Collection: Subjects
- subject_id: INTEGER
- Courses (Array):
  - course_id: INTEGER
...
Collection: Students
- student_id: INTEGER
- date_of_registration: DATETIME
...
MongoDB Data
{"Subjects":[{"subject_id":3,"subject_name":"Language",...},...]}

Prompt for Multilingual NLQs Synthesis in Data Augmentation

Role: SYSTEM
Content: You are a highly skilled multilingual translation assistant specialized in translating Natural Language Queries (NLQs) into different target languages. Your task is to accurately translate NLQs while maintaining consistency with the corresponding database queries (e.g., MongoDB queries) and ensuring the translations are natural and fluent. **During the translation process, you must not alter the representation of any MongoDB values mentioned in the NLQ**

Role: USER
Content:
#Given the Original Natrual Language Query List and Corresponding MongoDB Query, perform the following actions: ...
A: Let's think step by step!

Role: ASSISTANT
Content:
Let's break down the task step by step: ...

Role: USER
Content:
#Given the Original Natrual Language Query List and Corresponding MongoDB Query, perform the following actions:
1. Recognize the words that should be translated into Japanese.
2. Analyze the NLQ alongside its corresponding MongoDB query, identify the values corresponding to the query conditions within the natural language sentence, and preserve the consistency of these values' expressions as they appear in the MongoDB query throughout the subsequent translation of the NLQ.
3. Ensure the translation flows smoothly and naturally, with consistency in vocabulary.
4. Based on the steps above, Translate every NLQ in the List into the Target Language.
5. Output the translated NLQ in the following format:
```markdown
1. **Ori_NLQ1**: [Translated_NLQ1],
2. **Ori_NLQ2**: [Translated_NLQ2],
...
```
Original Natrual Language Query List:
```markdown
1. **Ori_NLQ1**: Retrieve the top 5 investors with the highest total transaction amounts along with their details.
...
5. **Ori_NLQ5**: Find the top 5 investors with the highest total transaction amounts and provide their details.
```
Corresponding MongoDB Query:
```javascript
db.Investors.aggregate([
 { $unwind: "$Transactions" },
 { $group: { _id: "$investor_id", Investor_details: { $first: "$Investor_details" }, totalTransactionAmount: { $sum: "$Transactions.amount_of_transaction" } }},...]);
```
A: Let's think step by step!

E.2.2 Retrieval-Augmented Chain-of-Thought Query Generation

Prompt in Retrieval-Augmented Chain-of-Thought Query Generation

Role: SYSTEM
Content: You are now the natural language interface for MongoDB, responsible for referencing the English NoSQL query examples provided by the user and generating MongoDB queries corresponding to the natural language queries based on MongoDB schemas.

Role: USER
Content:
Given the complete MongoDB Schemas, potential MongoDB fields to be used, natural language queries, and reference English NoSQL queries, please perform the following actions:
1. Analyze the natural language query based on the reference English NoSQL query;
2. Infer the MongoDB fields to be used based on the analysis from the first step, the complete MongoDB Schemas, and the potential MongoDB fields provided by the user;
3. Modify the reference English NoSQL query using the MongoDB fields inferred in the second step to make it suitable for the current MongoDB database;
4. Output the final NoSQL query in the following format:
```javascript
<final MongoDB query>
```

Retrieved Examples

Relevant Schema
...
Natural Language Query
...
Reference English NoSQL Query
...
Final MongoDB Query
...

...

Complete MongoDB Schemas
```markdown
### Collection: 学生
- 学生ID: INTEGER
- 姓: VARCHAR(12)
...
```
Relevant Schema
学生: _id, 专业, 性别

Natural Language Query
- `在招生人数方面，女生中领先的专业是什么？`

Reference English NoSQL Query
```javascript
db.students.aggregate([{$match: {gender: "female"}},{$group: {_id: "$major",count: {$sum: 1}}},{$sort: {count: -1}},{$limit: 1},{$project: {major: "$_id",_id: 0}}]);
```

Final MongoDB Query