SNAILS: Schema Naming Assessments for Improved LLM-Based SQL Inference

KYLE LUOMA, University of California, San Diego, USA and United States Military Academy - Army Cyber Institute, USA
ARUN KUMAR, University of California, San Diego, USA

Large Language Models (LLMs) have revolutionized Natural Language to SQL (NL-to-SQL), dominating most NL-to-SQL benchmarks. But LLMs still face limitations due to hallucinations, semantic ambiguity, and lexical mismatches between an NL query and the database schema. Naturally, a lot of work in the ML+DB intersection aims to mitigate such LLM limitations. In this work, we shine the light on a complementary data-centric question: *How should DB schemas evolve in this era of LLMs to boost NL-to-SQL?* The intuition is that more NL-friendly schema identifiers can help LLMs work better with DBs. We dive deeper into this seemingly obvious, but hitherto underexplored and important, connection between schema identifier "naturalness" and the behavior of LLM-based NL-to-SQL by creating a new integrated benchmark suite we call SNAILS. SNAILS has 4 novel artifacts: (1) A collection of real-world DB schemas not present in prior NL-to-SQL benchmarks; (2) A set of labeled NL-SQL query pairs on our collection not seen before by public LLMs; (3) A notion of naturalness level for schema identifiers and a novel labeled dataset of modified identifiers; and (4) AI artifacts to automatically modify identifier naturalness. Using SNAILS, we perform a comprehensive empirical evaluation of the impact of schema naturalness on LLM-based NL-to-SQL accuracy, and present a method for improving LLM-based NL-to-SQL with natural views. Our results reveal statistically significant correlations across multiple public LLMs from OpenAI, Meta, and Google on multiple databases using both zero-shot prompting as well as more complex NL-to-SQL workflows: DIN SQL, and CodeS. We present several fine-grained insights and discuss pathways for DB practitioners to better exploit LLMs for NL-to-SQL.

CCS Concepts: • **Information systems → Data management systems**; *Database views*; *Structured Query Language*; • **Computing methodologies → Natural language processing**.

Additional Key Words and Phrases: database; relational database schema; schema design; natural language to SQL; LLM; benchmark; schema naturalness; schema linking

ACM Reference Format:
Kyle Luoma and Arun Kumar. 2025. SNAILS: Schema Naming Assessments for Improved LLM-Based SQL Inference. *Proc. ACM Manag. Data* 3, 1 (SIGMOD), Article 77 (February 2025), 26 pages. https://doi.org/10.1145/3709727

1 Introduction

Natural language-to-SQL (NL-to-SQL) query generation capability has been revolutionized by foundational large language models (LLMs) [31, 42, 49]. This has made the integration of LLM-based query tools into relational database workflows more viable, with both established DBMS vendors and startups beginning to offer commercial NL-to-SQL interfaces. However, challenges in the NL-to-SQL space remain that can degrade the effectiveness of an LLM-enabled data retrieval

Authors' Contact Information: Kyle Luoma, kluoma@ucsd.edu, kyle.luoma@westpoint.edu, University of California, San Diego, La Jolla, California, USA and United States Military Academy - Army Cyber Institute, West Point, New York, USA; Arun Kumar, akk018@ucsd.edu, University of California, San Diego, La Jolla, California, USA.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2025 Copyright held by the owner/author(s).
ACM 2836-6573/2025/2-ART77
https://doi.org/10.1145/3709727

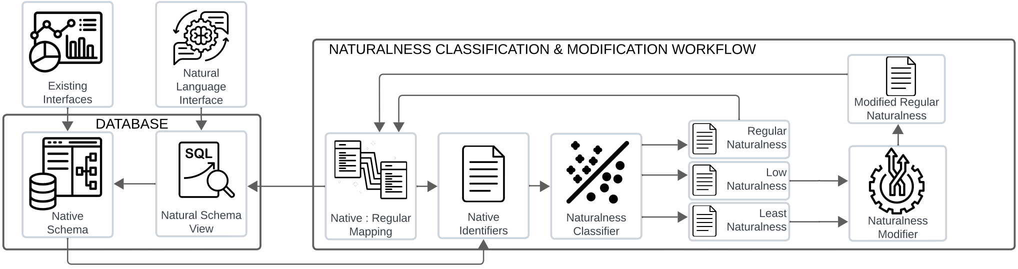

Fig. 1. Databases with poorly named, or less natural, schema identifiers perform poorly in LLM-based NL-to-SQL interfaces, and this project exposes the need for more natural schemas. We offer approaches and artifacts, including a naturalness classification and modification workflow, that can aid in the naturalness assessment and modification processes required to create a performance-enhancing natural view. In this way, the native schema remains as-is so that existing tools can continue talking to it without modification, while an LLM-based NLI can be integrated into the existing stack via a natural view.

workflow in real-world databases [11]. Principal among such challenges is *schema linking*, which is the association of entities in NL utterances to elements in the database schema.

While much work has studied making LLMs larger or more sophisticated, a more basic issue often underlies this challenge: lexical mismatches between natural language and poorly-named tables and columns in a schema. Intuitively, schema elements that are “better named” could raise the accuracy of schema linking within the NL-to-SQL setup. In this paper, we unpack and dive deeper into this intuition to study how exactly the “naturalness” of schema elements matters for NL-to-SQL by instituting a new benchmark and performing extensive empirical analysis using that. One might ask: *Why bother formalizing a concept that seems obvious and intuitive?* We believe this is important for 2 reasons. First, without a more formalized–or at least automated way–to define, verify, and compare “naturalness” researchers and practitioners alike will be forced to grapple with ad hoc and inconsistent approaches. In turn, this can lead to confounded conclusions by researchers on how different LLMs behave on different schemas and mislead practitioners comparing different NLIs. This points to the need for a new benchmark labeled dataset for this problem.

Second, practitioners need a way to efficiently and accurately operationalize any insights about the impact of naturalness on their schema elements for LLM-based NLIs. This points to the need for a systematic evaluation of how naturalness affects different databases, queries, and LLMs used for NL-to-SQL.

Our Focus. In this paper, we take the first steps toward deeper understanding on this seemingly obvious, but hitherto underexplored and important, relationship between schema identifier naturalness and LLM-based NL-to-SQL. Specifically, we ask the following three interconnected questions. (1) How do we quantify “naturalness” of schema identifiers? (2) Does it really impact schema linking accuracy in LLM-based NL-to-SQL and if so, by how much? (3) How does that impact vary by complexity of the database and query, as well as across different popular LLMs?

To answer the above questions, we create a novel integrated benchmark suite we call SNAILS with new collections of real-world databases and query pairs, a new labeled dataset of schema identifiers, a set of evaluation metrics, and LLM prompting and other AI artifacts.

1.1 Preliminaries and Setup

LLM-based NL-to-SQL. The most obvious way to seek LLM performance improvements would be by increasing the power of the language models themselves. But the cost of training and deploying LLMs continues to increase in concert with their complexities. Additionally, many practitioners seek "plug and play" solutions by employing already-available LLMs. Model training and finetuning impose access barriers that may render such a pursuit untenable for organizations that use databases but lack the requisite talent such as data science and machine learning expertise.

The practice of prompt engineering can also help improve NL-to-SQL performance, though dealing with schema complexity and schema representations in LLM prompting is an ongoing challenge in enterprise-level NL-to-SQL applications [11]. The majority of leading submissions on the popular Spider NL-to-SQL benchmark leaderboard are LLM-based solutions [8, 13, 40] that employ a variety of prompting strategies, some of which require multiple successive API requests containing schema context and instructions. These approaches can be costly and unintuitive for NLIDB end users, and can incur excessive costs and overhead when deployed at scale.

A complementary line of work on realistic NL-to-SQL benchmarking uses structural schema modification such as normalization, flattening, and replacement to evaluate effects on LLM performance. Making such structural changes to target schemas challenges model robustness and increases error rates in NL-to-SQL performance [27], and this recent work indicates that schema design is a viable target for LLM-based NL-to-SQL accuracy improvements.

Schema Linking. Schema linking remains as a persistent challenge for LLMs. With the availability of capable LLMs that consistently generate valid SQL statements, a larger proportion of NL-to-SQL generation errors are now associated with incorrect or ambiguous database identifier selection as opposed to incorrect syntax [46]. Schema linking performance has been improved using lexical matching heuristics [16, 56], joint relationally aware embeddings with attention [3, 52], the use of pre-trained language models to perform schema probing [53], and multimodel pipelines with ML models for pruning schema knowledge [22]. Some NL-to-SQL methods address schema linking challenges by adding additional context such as sample values or metadata [40] to schema knowledge representations. These methods can improve performance in some cases [29], and can be useful for schemas with obscurely-named tables and columns, though they do so at the cost of much larger schema knowledge representations.

Schema linking still often fails, even with the most capable LLMs due to poorly-aligned schema identifier names with natural language question contents, that could be due to the use of synonyms or the obscurity of a database identifier. In the latter case, it can be challenging for even a sophisticated linking solution to match natural language words to schema elements that yield minimal semantic meaning.

Schema Naming Conventions. The majority of database schema naming best practices originate from *practitioners* and are generally published as software documentation, organization policies, tutorials, etc. We find that there is a gap in database and data integration *academic* literature evaluating schema identifier naming practices for any purpose. While the semantics of schema identifiers may not have been considered as a necessary subject of database research in the past, the increasing integration of natural language interfaces to databases has elevated its importance.

Naming conventions for database schema identifiers vary by organization, database vendor, application, and purpose. A web search for database table and column naming guidelines yields multiple resources ranging from blog posts [4], StackOverflow responses [44], DBMS vendor documentation [33], and tutorials [15]. Poor schema identifier naming practices is considered a database code smell [43] where meaningless identifier names should be avoided. Generally, the most consistent best practices include selecting descriptive and concise names that contain only commonly-understood abbreviations and acronyms, though some conventions suggest the use of abbreviated prefix and suffix modifiers that describe application associations, or entity purpose [34].

In our research, we identified several databases containing schemas with varying levels of human-readability and understandability (what we will call naturalness) which suggests that there can be a tendency for database schema designers to choose shorter and less descriptive identifier naming conventions. As we will see, such naming shortcuts can negatively affect NL-to-SQL performance.

1.2 Our Benchmark Artifacts and Analyses

Given the above context of our benchmarking setup, we now explain the new artifacts in SNAILS, followed by a summary of our empirical analysis.

Artifact 1: Real-World Database Schemas. The SNAILS benchmark contains several new *real-world database schemas* that are not part of existing NL-to-SQL benchmarks (Artifact 1). Our focus on schema naming motivates the creation of a new novel benchmark dataset, because existing benchmark naturalness levels are higher than those of many real-world schemas, and other real-world schema collections including SchemaPile [7] lack the necessary database instances to enable NL-to-SQL evaluation. In our analysis of these real-world schemas, we discover that identifier naming variances generally appear in the form of abbreviations and expansions; we refer to these variances as identifier *naturalness*.

Artifact 2: Identifier Naturalness Classifications. Our analysis reveals that naturalness can be formalized categorically with the help of finetuned language models and feature engineering. We then hand-label the schema identifiers, with some ML assistance, to classify their naturalness level and produce a new golden labeled dataset. We classify identifiers into one of 3 naturalness levels (Regular, Low, and Least) (Artifact 2). This dataset, consisting of over 17,000 labeled identifiers, serves as the training data for the naturalness classifiers described next.

Artifact 3: Naturalness Classifiers. We experiment with various classification approaches, and make available the models trained to classify the naturalness of a database schema identifier (Artifact 3).

Artifact 4: Naturalness-Modified Identifiers. To better understand the effect of schema identifier naturalness, and to enable within-database experiments, we create alternate versions of each real-world schema identifier at each naturalness level (Artifact 4). This dataset serves two purposes: 1) Training data for ML-based naturalness modifiers, and 2) Generation of schemas with varying naturalness levels to analyze the impact of naturalness on NL-to-SQL performance. We modify the identifiers with the assistance of LLM prompting, finetuned models, and database metadata.

Artifact 5: Naturalness Modifier. We offer an in-context learning-based prompting strategy for identifier naturalness reduction (or abbreviation). We also provide an identifier naturalness increaser (or expander) that leverages retrieval augmented generation, interactive few-shot example building, and database metadata parsing methods to streamline the database naturalness improvement process.

Artifact 6: NL-to-SQL Question Query Pairs. The SNAILS benchmark contains 503 NL question-SQL query pairs which we use for NL-to-SQL performance analysis of 4 LLMs. We created this new collection as another hand-labeled golden dataset without the use of AI-based workflows (Artifact 6).

Experimental Evaluation. Using the SNAILS benchmark artifacts, we analyze and experiment with the effects of schema identifier naturalness on LLM NL-to-SQL performance. We select 5 publicly-available LLMs: OpenAI's GPT-3.5, GPT-4o, a finetuned variant of Meta's Code-Llama, Google's newest Gemini 1.5, and CodeS finetuned for NL-to-SQL. We evaluate them using both execution result set matching and a novel identifier set comparison approach that pinpoints schema linking performance.

In this paper we focus primarily on a simple zero-shot prompting of the LLM for our experiments. We recognize that this may not be the best for overall execution accuracy, but it helps us isolate the impact of schema identifier naturalness in this first work on this problem. As such, more complex workflows will create confounding effects while not necessarily providing more insights into schema linking performance. However, for completeness sake, we also compare two illustrative complex workflows: DIN SQL for task-specific prompt chaining [40], and CodeS [23] for NL-to-SQL finetuning.

We find that schema identifier naturalness by and large does have a meaningful effect on NL-to-SQL accuracy and schema linking performance. Specifically, identifier naturalness is moderately and positively correlated with both schema linking and execution accuracy. Identifiers of low naturalness yield lower performing NL-to-SQL inferences in terms of both schema linking (identifier recall) and execution accuracy. These findings have implications for practitioners who are either designing new databases intended for LLM-based applications, or seeking to augment existing RDBMSs with an LLM-based NL-to-SQL interface.

In summary, this paper makes the following contributions:

- We propose a novel measure of *naturalness* of a database schema identifier and demonstrate through extensive experiments that naturalness has a significant effect on LLM schema linking performance in the context of NL-to-SQL.
- We provide a hybrid LLM-generated and human-curated training dataset (Artifact 2) and language model (Artifact 3) for schema naturalness classification.
- We offer a new multi-domain NL-to-SQL evaluation benchmark collection consisting of 9 real-world relational databases (Artifact 1) and 503 unpublished NL-to-SQL query pairs (Artifact 6) that do not exist in any LLM training corpora.
- We create a novel labeled dataset of alternate naturalness levels that map the identifiers from Artifact 1 to hybrid LLM-human curated identifiers of different naturalness levels (Artifact 4), and methods for expanding and abbreviating identifiers to change their naturalness (Artifact 5).
- We conduct an extensive empirical analysis of the performance of 5 popular foundational LLMs over our benchmark using a novel schema linking metric for NL-to-SQL.
- We propose a realistic workflow that enables the preservation of existing database integrations while offering LLM-based NLIs a natural view of a target schema.

2 Schema Identifier Naturalness

Intuitively, naturalness can be thought of as the degree to which a phrase, or word, resembles natural language. Naturalness is a concept and target of research in field of controlled natural languages [21], where controlled language syntax is evaluated in terms of naturalness levels. Recent NL-to-SQL research also defines and measures naturalness [27] for the purpose of evaluating the naturalness of natural language question utterances, but avoids measuring the naturalness of schema elements.

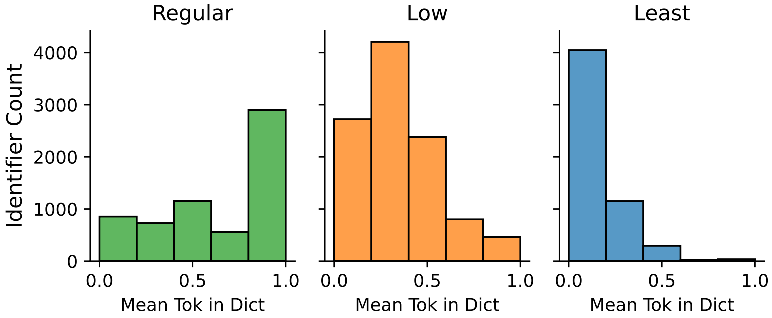

Fig. 2. *Mean Token in Dictionary*, the proportion of tokens in an identifier that match a word in an English dictionary, generally aligns with the SNAILS 3-class naturalness categorization approach.

Regular	Low	Least
airbag	AccountChk	AdCtTxIRWT
AdaptiveCruiseControl	IsueFrDate	COGM_Act
ModelYear	RecvAsst	DfltSlp
service_name	UsrQuery	FNDAbs
Research_Staff	ValueOfT	CSI22

Table 1. Example identifiers and their naturalness levels, from the SNAILS naturalness labeled dataset (Artifact 2).

To the best of our knowledge, no prior attempts have been made to definitively measure the naturalness of a database schema's identifiers. In order to achieve this goal, we propose a three-category naturalness classification scheme in order to measure the effects of naturalness on NL-to-SQL performance.

2.1 Naturalness Categories

As the first work on this topic of how schema identifier naturalness affects LLMs, we seek to define a preliminary metric–one that is consistent and descriptive enough to differentiate between naturalness levels and to measure their effects.

To gain insights into naturalness-related trends in the SNAILS datasets, we create a *mean token-in-dictionary* measurement that describes the proportion of tokens in an identifier that exactly match a word in a comprehensive English word list. Figure 2 reveals differences between each naturalness category where Least naturalness identifiers contain fewer in-dictionary tokens, and Regular naturalness identifiers are more likely to consist of in-dictionary tokens. This distribution suggests that because the bulk of the training corpora of LLMs is human-generated natural language text, what humans consider "natural" for such identifiers generally aligns with how LLMs react to them.

Examples of schema identifiers and their naturalness categories are displayed in Table 1. We define these categories with the underlying assumption that the identifiers are named as some semantic representation of the data, and that naming-related problems of interest are related how an identifier is codified. That is, identifiers are assumed to not be random character sequences or random words that do not correspond to the content of the database entities they represent. With this assumption in mind, we categorize naturalness into 3 discrete levels as follows:

- Regular: The identifier contains complete English words with no abbreviations or acronyms, or contains only acronyms in common usage (e.g., ID or GPS).
- Low: The identifier contains abbreviated English words and less common acronyms that are usually recognizable by non-domain experts (e.g., UTM or CPI). The meaning of the identifier can be inferred without consulting external documentation.
- Least: The identifier's meaning cannot be inferred by non-experts due to indecipherable acronyms or abbreviations, and external metadata or other documentation must be consulted in order to determine its purpose.

While we recognize that naturalness can also be treated as a continuous spectrum, between the choices of continuous scoring and discrete categories, we select the latter as an initial approach to naturalness evaluation. The primary factors underlying this choice are the level of effort required to conduct human-based scoring of a large set of database identifiers, and the difficulty of consistently scoring naturalness on a continuous range over a large set of data. Therefore, we use an intuitive and easily-verifiable discrete 3-class taxonomy in the first work on this topic.

2.2 Naturalness Classification

To consider naturalness as a factor in NL-to-SQL performance, we derive naturalness scores of the target schemas' identifiers. We use this score to consider effects of individual identifier naturalness, schema naturalness, and query identifier naturalness. Because manual naturalness classification can be a time consuming task for large schemas, we automate the process by training a machine learning-based classifier. This effort is beneficial in multiple situations. First, it can ease some manual effort of the labeling process and make the process of scaling to more databases in the future less labor intensive. Second, it can help practitioners efficiently and consistently evaluate the naturalness of their own database schema identifiers prior to NLI integration.

To train a classifier to perform identifier naturalness scoring, we employ the 3-class set of naturalness categories described in Section 2.1, and a list of database identifiers drawn from the SNAILS real-world database schemas (Artifact 1). We categorize the naturalness of each identifier to generate the SNAILS *identifier naturalness classification* labeled data (Artifact 2) which we use for ML-based naturalness classifier training, evaluation and testing.

We evaluate multiple classification approaches including heuristic-based word matching, few-shot LLM prompting with GPT-3.5 and GPT-4, and LLM finetuning. The GPT-4 few-shot approach achieves 74 percent accuracy and an f1 score of 0.77. We experiment with multiple finetuning collections, first using a hand-labeled collection of 1,648 naturalness classifications and then leveraging the initial classifier along with weak supervision to generate a larger collection of 17,226 labeled identifiers. Finetuning using the second collection outperforms all few-shot approaches, with the two best-performing classifiers fine-tuned GPT 3.5 and BERT-based CANINE [5] models performing at 89 percent accuracy, and 0.89 f1 score.

Figure 3 provides a visual comparison between the SNAILS schema collection and common NL-to-SQL benchmarks including Spider, Spider Realistic, and BIRD. Additionally, we compare the SNAILS collection to the real-world SchemaPile collection and find that SNAILS collection proportions generally align to SchemaPile naturalness, more so than other existing benchmarks, which creates a more realistic and challenging benchmark in terms of schema naturalness.

To better understand the magnitude of naming practices in real-world schemas, we use the CANINE-based classifier to classify the naturalness of the SchemaPile collection: a large volume of real-world database schemas [7] that contains over 22,000 database schemas, 198,000 tables, and 1 million columns. We find that in over 7,500 schemas (32 percent of the collection) Least natural identifiers make up at least 10 percent of the schema identifier names. Additionally, over 5,000

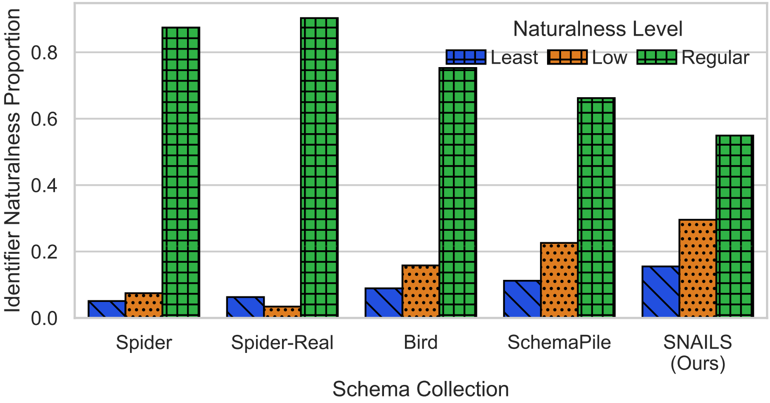

Fig. 3. Comparison of the SNAILS database collection (Artifact 1) described in Section 3.1 to other real-world and benchmark schema collections. SNAILS naturalness proportions are generally biased toward less natural identifiers and is more consistent with the real-world SchemaPile collection than other existing benchmarks including Spider and Spider Realistic.

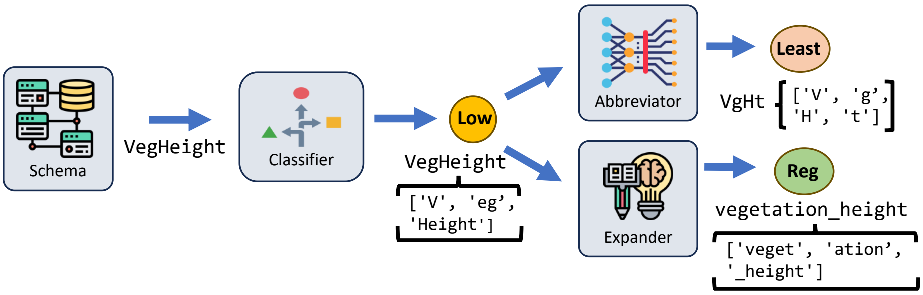

Fig. 4. Schema identifiers are classified (Artifact 2) and modified to increase or decrease naturalness as appropriate. Modified identifiers comprise the schema crosswalks used for schema modification during NL-to-SQL experimentation (Artifact 3).

schemas register a combined naturalness of 0.7 or below–an indicator that the schema contains a high level of Low and Least naturalness identifiers. We examined the naturalness category distribution for these 5,000 schemas, and found that for this subset of schemas Low and Least naturalness identifiers outnumber Regular naturalness identifiers. These findings reinforce the importance of the naturalness problem by confirming that, although a reasonable majority of schemas are already natural, there still exist many schemas with lower naturalness levels in the real-world–enough to motivate the formalization of schema naming quality measures.

2.3 Identifier Schema Naturalness Mapping

In addition to measuring the effects of identifier naturalness in existing schemas, we also seek to evaluate the effects of modifying schema naturalness. For this purpose, we create Artifact 4, naturalness-modified identifiers. This artifact enables schema modification during prompt generation and query inference, which provides a within-schema assessment of naturalness level effects on NL-to-SQL accuracy.

Identifier Mapping. In addition to the ground truth, or Native, naturalness of the 9 schemas in the SNAILS real-world database collection, the naturalness-modified identifier collection contains 3 additional sets of identifiers: Regular, Low, and Least. That is, each native identifier is mapped to 2 additional, semantically equivalent, identifiers of higher or lower naturalness, and mapped to itself for its own naturalness level (i.e., we do not generate new identifiers of the same naturalness as its native form).

Figure 4 provides a visual example of the Native identifier *VegHeight* which is classified as Low naturalness. With this naturalness classification as a starting point, we abbreviate it further to generate a corresponding Least naturalness identifier *VgHt*. We also expand it to generate the corresponding Regular naturalness version *vegetation_height*. We map the Native *VegHeight* identifier to itself in the Low naturalness category.

Naturalness Modification. For *more natural to less natural* modifications (the abbreviator in Figure 4), we employ in-context learning (few-shot) prompt strategies with GPT-3.5 turbo to generate naturalness-modified identifiers (e.g., Regular to Low, Low to Least, and Regular to Least). We favor this approach over model finetuning, as simple instructions to abbreviate the identifier coupled with several examples prove more effective and less prone to poor results (e.g., presence of unwanted characters in the modified identifier).

Automating the reverse *less natural to more natural* naturalness modification (the expander in Figure 4) requires additional context and external knowledge from data description sources. Though a recent project describes a promising identifier expansion strategy [57] without external knowledge, it requires finetuning over a large dataset, and is likely susceptible to overfitting; therefore we opt for our own approach that incorporates the use of an LLM augmented with schema metadata lookup capability. To accomplish this, we create a Python program with GPT interaction that takes as input metadata describing a schema's native identifiers, and outputs an identifier with regular naturalness. More details of this process are available in the technical report [26].

3 Base Collections

Given the recency of the LLMs selected for evaluation in this project, and the relative maturity of existing NL-to-SQL benchmarks, we believe that foundational LLMs have been exposed to existing benchmark training and development NL questions and queries in their training corpora. NL-to-SQL performance differences between queries over seen vs. unseen schema are significant [46], and we seek to avoid as much bias as possible due to intentional or unintentional pre-training on existing benchmark datasets.

We also find that existing benchmarks including Spider [55], and BIRD [24], do not match the identifier naturalness distribution of real-world schema collections such as SchemaPile [7]. Although SchemaPile is a very large representation of real-world schemas, it does not contain database instances necessary for benchmark performance evaluations; so, we are not able to leverage its dataset in the creation of a new benchmark. To reduce bias due to benchmark data exposure, and to create a benchmark more representative of real-world schema naming, SNAILS contains two artifacts for NL-to-SQL benchmarking: Artifact 1, which is a collection of 9 publicly-available database schemas and data; and Artifact 6, a human-generated set of 503 NL question - gold query pairs.

Database	Tables	Columns	Questions	Org
ASIS	36	245	40	NPS
ATBI	28	192	40	NPS
CWO	13	71	40	NPS
KIS	18	157	40	NPS
NPFM	27	190	40	NPS
NTSB	40	1611	100	NHTSA
NYSED	27	423	63	NYSED
PILB	21	196	40	NPS
SBOD	2588	90,477	100	SAP

Table 2. SNAILS Real-World Database Schemas

3.1 Datasets

Native Schemas. The SNAILS real-world database schema collection (Artifact 1) consists of 9 databases sourced from multiple locations. We refer to the schema identifier names as they exist in the source databases as *Native*, and we classify each schemas' Native naturalness level (see Figure 5). Domain diversity facilitates a more thorough evaluation [10]; so, SNAILS database collections span multiple domains. Domain coverage includes scientific nature observation records, vehicle safety statistics, primary school performance data, and business resource planning.

The U.S. National Parks Service's IRMA Portal [1] is the source of the scientific observation databases which include the Field Data for the Inventory of Amphibians and Reptiles of Assateague Island National Seashore (**ASIS**) [6], Great Smoky Mountains All Taxa Biodiversity Inventory (**ATBI**) Plot Vegetation Monitoring Database [9], Wildlife Observations Database: Craters of the Moon National Monument and Preserve 1921-2021 (**CWO**) [45], Exotic and Invasive Plants Monitoring Database (**KIS**) [19], Northern Plains Fire Management (**NPFM**) [28] and Pacific Island Network Landbird Monitoring Dataset (**PILB**) [20].

The National Transportation Safety Bureaus 2021 safety sampling dataset [30, 41] is the source of SNAILS **NTSB** safety statistics database. We source school performance data (**NYSED**) from the New York State Education Department [2].

The business resource planning database **SBOD** is a training example of the popular SAP Business One system, and is publicly available in MS SQL server backup format [39]. The SBOD schema consists of an extremely large number of tables and columns; so pruning is required to fit the schema within the context window of the LLMs we compared. We reduce the schema knowledge token requirements by segmenting the SBOD schema into submodules and further reducing tables through data profiling. Additional information on the SBOD schema knowledge management is available in the technical report [26].

Each database was migrated from its source format into an MS SQL Server database. Several databases contained identifiers with whitespace characters, which is uncommon in most schemas. To mitigate whitespace-related inference failures as a confounder, we modify the native identifiers by replacing whitespace characters with underscore characters. In total, 148 out of over 19,000 total identifiers (less than .01 percent) contained at least 1 whitespace character.

Native Schema Naturalness Levels. Since the intent of this project is to measure the effect of schema naturalness, we first check if there is sufficient distribution of naturalness levels across the collection. We employ the GPT-3.5-based classifier described in Section 2.2 to evaluate the naturalness of the native schema identifiers.

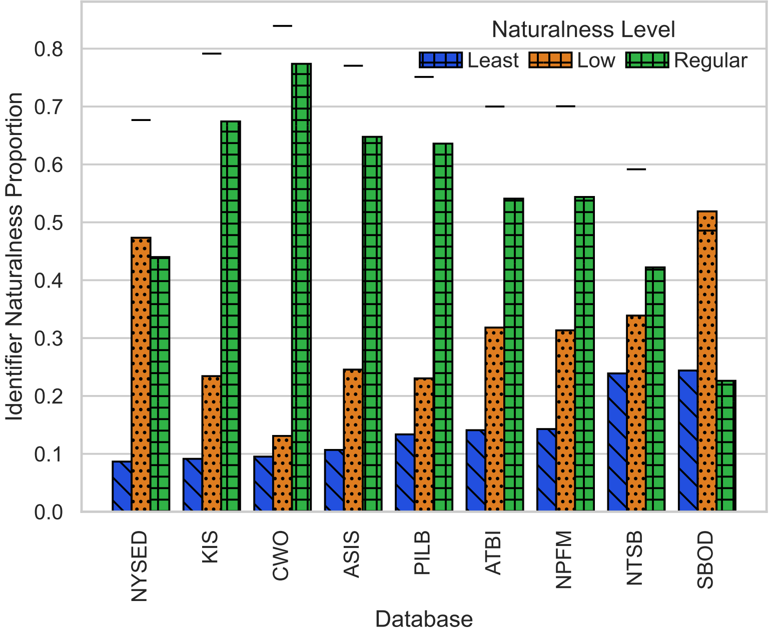

Fig. 5. Proportion of identifiers in each naturalness category within the SNAILS real-world database collection (Artifact 1). Horizontal line markers indicate calculated combined naturalness as described in the technical report [26]

In addition to measuring the proportion of identifiers in each naturalness category, we also derive a combined naturalness score. Combined naturalness is the weighted average of category proportion values, where scores range from 0.0 to 1.0 with 1.0 representing a schema containing only Regular naturalness identifiers. A more detailed description of its calculation is available in the technical report [26].

Figure 5 displays the proportions of identifiers in each naturalness category, as well as the combined naturalness, in each native schema. From the chart, we can see that the schemas in the SNAILS collection described in Section 3.1 represent a heterogeneous selection of naturalness combinations.

Modified (Virtual) Schemas. To control for confounding factors such as schema structure, normalization levels, and constraint variances between native schemas, we perform within-database evaluations of naturalness. To accomplish this, we generate 3 additional *virtual* schemas using the naturalness-modified identifiers (Artifact 4) described in Section 2.3. Each virtual schema is representative of a naturalness category, where schema identifiers are replaced with a semantically equivalent identifier of a different naturalness level. This results in 4 schema versions per database in the base collection: Native, Regular, Low, and Least.

The modified schemas are virtual because we do not create database instances that can be queried directly. Rather, we query virtual schemas via identifier replacement in prompts and generated queries using processes described in Section 4. This approach reduces storage overhead. It also enables possible future schema variations of different naturalness proportions without the need to instantiate additional database instances.

Database	Qs	Top	Funct.	Join	C-Join	Ex	SQ	Where	Neg	Grp	Ord	Hvg
ASIS	40	1	24	13	1	0	2	18	0	17	1	0
ATBI	40	5	20	18	0	1	7	21	2	16	7	1
CWO	40	2	18	5	1	5	10	34	7	12	2	1
KIS	40	8	26	15	0	0	2	25	1	11	8	0
NPFM	40	5	27	21	0	0	1	29	0	16	5	0
NTSB	100	8	82	23	21	0	6	62	4	42	23	4
NYSED	63	10	36	10	4	1	21	55	1	16	10	1
PILB	40	6	25	23	0	0	3	20	0	16	11	2
SBOD	100	2	33	44	0	0	0	82	0	17	2	1

Table 3. Gold query clause counts for each SNAILS database. Columns represent a count of gold queries that contain the listed clause types. Qs is the count of question-query pairs for each database. C-Join is the subset of joins that require a composite key. Ex indicates the use of an exists clause. SQ indicates a subquery. Neg, Grp, Ord, and Hvg represent negation, group by, order by, and having. Note: MS SQL Server dialect replaces the common LIMIT clause with an equivalent TOP clause that precedes select items in the SELECT clause.

SNAILS Database Selection and Extension Processes. The initial 9 datasets and schemas are included because they were (1) publicly available, (2) not included in any prior NL-to-SQL benchmarks, (3) contained relational tables with dependencies and database instances with values, (4) had available table and column metadata, (5) represented a diversity of application domains, and (6) contain data potentially useful for real-world data analysis or data science applications. Databases are not selected or pre-screened using perceived naturalness as criteria.

We view the initial 9 schemas as a starting point from which the SNAILS dataset can grow. Researchers who wish to extend the SNAILS collection should use the same selection criteria. In addition, the extension process must ensure that new databases: (1) can be represented as MS SQL Server instances, (2) each native identifier's naturalness is classified according to defined criteria using the SNAILS naturalness classifier, and (3) that native identifiers are modified using the SNAILS modification artifacts to create alternate naturalness levels.

3.2 NL Question - SQL Query Pairs

To evaluate SQL inference performance over the Native and modified schemas in the SNAILS real-world database collection, we create a new set of 503 NL-question and SQL gold query pairs (Artifact 6). Schema identifier naturalness are the primary considerations for NL question and gold query composition. During question and query formulation we track schema coverage to ensure that the distribution of identifier naturalness within a set of gold queries generally matches the naturalness distribution of target schemas.

To enable accuracy measurements at the identifier level, gold queries contain the minimum identifiers (tables and columns) required to answer its corresponding question. For this reason, for questions that require the count aggregation function, where appropriate, we use the COUNT(*) clause (as opposed to selecting an arbitrary column). This approach eliminates incorrect penalties to recall if a generated query fails to project an arbitrary column as a function argument.

Gold queries contain only native identifiers, such that all gold queries return valid non-null results from target databases in the real-world database collection (Artifact 1). We measure query complexity as a count of its clauses and identifiers. Gold queries span a range of complexities, from very simple single table projections, to multi-table joins and nested subqueries (see Table 3).

Adding New NL-SQL Pairs to the SNAILS Collection. For researchers interested in extending the SNAILS collection, it is necessary to create new ground truth NL-SQL pairs for evaluation. While we employed a fully manual approach for question writing, and this approach may be used for future additions, they may also consider the use of new approaches such as using a template-based approach for generating question-query pairs with relational data as input [36]. Regardless of NL-SQL pair creation method, researchers should ensure adequate schema coverage and minimum essential identifier selection as described in the preceding section.

4 NL-to-SQL Benchmarking Setup

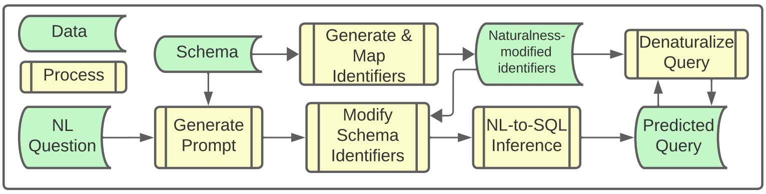

Fig. 6. Experiment setup workflow from NL question and schema as input to predicted query as output.

To evaluate the impact of naturalness on NL-to-SQL accuracy, we build a benchmarking setup pipeline as shown in Figure 6. NL question and gold query pairs, database schemas, and naturalness crosswalk mappings are inputs into subprocesses. The subprocesses include prompt generation, schema identifier naturalness modification, identifier naturalness classification, LLM-based NL-to-SQL inference, and predicted query "denaturalization" (i.e., converting table and column identifiers to native schema identifiers prior to query execution). The output of the experiment setup is a predicted query, which along with its gold query counterpart, is executed against a target database. This predicted query is passed into a parser analysis tool as initial steps of the *Performance Evaluation and Results Classification* phase of the experiment described in Section 5.

4.1 Prompt Generation

The design space for LLM-based NL-to-SQL prompting is quite large, with options ranging from zero-shot instructions to sequential prompting broken into discrete tasks such as schema subsetting and error handling. Although we evaluate 2 complex NL-to-SQL workflows, to maintain consistency across the LLMs compared in this study, our performance comparisons focus on a single prompting strategy: zero-shot prompting with schema knowledge.

Prompting Strategy. SNAILS prompts consist of zero-shot instructions with schema knowledge (denoted as ZS in results figures) in a format similar to OpenAI's Text-to-SQL demonstration prompt [13] for completions. The prompt begins with task instructions and database information:

```
For the database described next, provide only a sql query.
do not include any text that is not valid SQL.
#Database: NTSB
#MS SQL Server tables, with their properties:
```

Target database system tables provide schema knowledge, which is represented as a list of tables and their column names with data types in the format:

```
#TableName (Col1Name Type, Col2Name Type, ...)
```

The prompt ends with the instruction:

```
### a sql query, written in the MS SQL Server dialect,
    to answer the question: <Question>
```

Where <Question> is replaced with an NL question directed at the given schema.

To evaluate naturalness effects on more complex NL-to-SQL prompting workflows, we also implement DIN SQL [40] which uses prompt chaining with GPT-4, and CodeS [23]–a multi-step NL-to-SQL system (schema filtering and SQL inference) based on StarCoder [25] and finetuned for the NL-to-SQL translation task.

Prompt Schema Identifier Modification. For inference on virtual schemas with modified naturalness levels, we replace Native identifiers with corresponding identifiers of the target virtual schema's naturalness level. We accomplish this step using the naturalness-modified identifier collection (Artifact 4) described in Section 2.3. We use a SQL parser to encase identifiers within tags to improve identifier replacement accuracy and eliminate errors due to substring matching between identifiers.

4.2 NL-to-SQL Inference

Language Models. Foundational LLMs continue to grow in capability at a rapid pace. Despite this growth, not all NLI implementations can avail of the most-capable LLMs, often due to organizational policy constraints (e.g., organizational security concerns [14]). Additionally, we seek to understand if schema naming effects generalize across model architectures and sizes. Thus, we consider several LLMs, both open and closed source, to capture as many use profiles as possible including OpenAI's GPT-3.5 Turbo and GPT-4o [31, 32]; Google's Gemini 1.5 Ultra [47, 48]; and Phind-CodeLlama-34B-v2 [38] which is a finetuned variant of Meta's CodeLlama 2 [42].

CodeS and DIN SQL Implementation. For the more complex DIN SQL and CodeS NL-to-SQL workflows, we provide additional versions of the SNAILS schema artifacts to conform to the input requirements of the target systems. Additionally, we add data logging between agents to document the schema filtering step for additional analysis. For consistency between approaches, we use GPT-4o for all steps in the prompting chain. For CodeS inference, we execute the schema filtering and NL-to-SQL inference using the CodeS codebase and finetuned models.

Generated Query Denaturalization. For queries targeted at virtual schemas and generated using modified schema identifiers, we perform reverse modifications prior to query execution on the native database schema. Using a purpose-built Antlr [37]-based parser, we extract table and column identifiers, and generate a tagged query with identifier tags encasing table and column names. The tags guide the replacement algorithm, ensuring accurate replacement of naturalness-modified identifiers with their Native naturalness counterparts.

5 NL-to-SQL Benchmarking Results

This section describes the process of evaluating the generated SQL query output from the prior section. We evaluate performance in terms of execution accuracy (result set comparison and manual evaluation) and schema linking (recall, precision, and F1).

Key Takeaways. Overall, there is a model-dependent statistically significant correlation between identifier naturalness and execution accuracy, with smaller models exhibiting higher correlations between naturalness and performance. The presence of Least naturalness identifiers has the largest negative effect on schema linking. Additionally, while the performance difference between Regular and Low is visible, it is less impactful. So, modifying Least naturalness identifiers should be a higher priority than modifying Low naturalness identifiers.

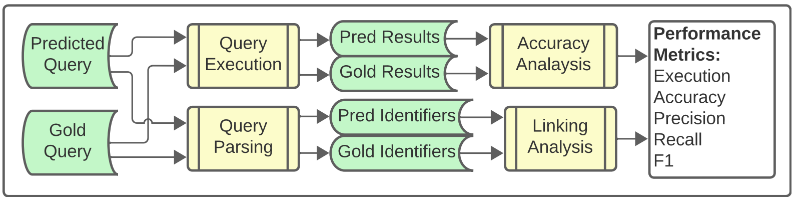

Fig. 7. Benchmark results evaluation includes generated and gold query execution on target schemas, parser-based analysis, and identifier set comparisons. We evaluate performance in terms of execution accuracy and schema linking (precision, recall, and F1).

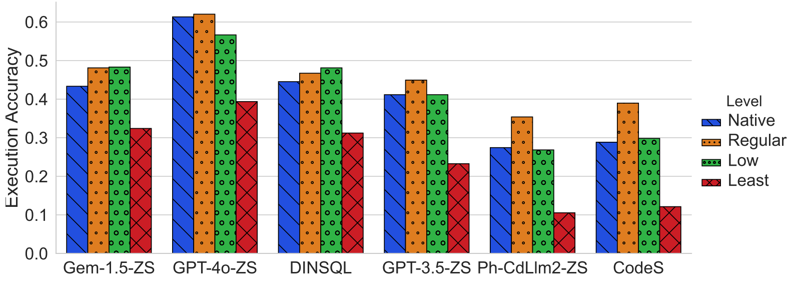

Fig. 8. Execution accuracy (proportion of correct queries) by model. There is slight accuracy improvement from native schemas to schemas modified to regular naturalness. Accuracy drops significantly for schemas modified to low naturalness.

5.1 Execution Accuracy

Execution Result Set Comparison. Execution accuracy is the standard measure of performance in most NL-to-SQL benchmarks [24, 55] where accuracy is determined using result set comparisons between gold and generated queries executed over one or more database instances. A drawback of existing methods is that strict set or bag comparisons risk increased false-negatives when a generated query includes additional fields that are not required, but do not render the result incorrect in terms of the natural language question [11, 56].

To reduce false negatives, the SNAILS approach to execution accuracy evaluation adopts 2 aspects of relaxed execution matching as described in [11]; it accounts for: (1) The possibility that a predicted query may contain additional columns beyond those retrieved by a gold query; and (2) That unless specified in the NL question, tuples may appear in any order. To achieve this, we perform result set-superset comparisons to ensure that the predicted result set column set is a superset of the gold result set column set. That is, a generated query is considered incorrect if it does not contain *all* gold query columns; but it is not considered incorrect (at this stage) if it includes columns not present in the gold query result. A more detailed description of this approach is available in the technical report [26].

Manual Evaluation. Execution result set comparison cannot prove query correctness; so we rely on it only to rule out true negatives from further consideration. To validate correctness, the authors manually review generated queries that pass execution result set-superset comparison checks. We streamline this process by creating a Python-based manual validation user interface that makes the process of comparing gold and generated queries more user-friendly. Manual validation steps include ensuring the generated query answers the NL question, matches the gold query in terms of semantic structure, and does not contain semantically incorrect predicates, projections, or clauses.

Naturalness Effect on Execution Accuracy. Figure 8 shows execution accuracy for each LLM and naturalness level. There is a clear difference in overall performance between LLMs, most likely due to model size. We find that generally more natural database schemas yield more correct queries. Databases with more natural native schemas did not benefit from identifier renaming, though we observe that altering a schema to become less natural degrades accuracy in most cases. We find that for databases with Native schema combined naturalness scores less than 0.69, modifying the schema identifiers to increase naturalness improves execution accuracy.

Statistical Significance. The Kendall-Tau correlation between the naturalness of identifiers in a query and execution accuracy ranges from low ($\tau = 0.09, p < 0.0001$) for Gemini 1.5, to moderate ($\tau = 0.19, p < 0.0001$) for Phind-CodeLlama2 and CodeS. The most impactful relationship is between the presence of Least naturalness identifiers and performance, with Kendall-Tau correlations between the proportion of Least identifiers in a query and execution accuracy between $\tau = -.15$ and $\tau = -.22$ with $p < 0.0001$ for all models.

5.2 Schema Linking Evaluation

We make schema linking a “first class citizen” of our analysis, and study schema linking performance in queries irrespective of other aspects of correctness. Thus, we propose query-level and identifier-level schema linking measurements. We propose an approach similar to the Spider benchmark exact set matching system [55] in which we employ a schema linking-specific evaluation method using *recall* scoring of gold and generated query pairs. Other schema linking-focused research measure effects of schema linking improvements using ablation [3, 46, 52, 53]. In other cases, schema linking is described in post-hoc analysis of NL-to-SQL model performance, with schema linking accounting for roughly 30% of failures [8, 40].

Query-Level Linking Analysis. The set of all schema identifiers (table and column names) present in gold queries represents the minimum identifiers required to correctly answer an NL question. Our purpose-built ANTLR4-based [37] query parser extracts identifiers from gold and generated queries. With a set QI_g of identifiers present in the gold query and a set of identifiers QI_p present in the generated (or predicted) query, we calculate recall, as well as F1 and precision.

$$QueryRecall = \frac{|QI_g \cap QI_p|}{|QI_g|} \tag{1}$$

$$QueryPrecision = \frac{|QI_g \cap QI_p|}{|QI_p|} \tag{2}$$

$$QueryF1 = \frac{2(QueryRecall * QueryPrecision)}{QueryRecall + QueryPrecision} \tag{3}$$

We exclude 137 linking score calculations from analysis in situations where the predicted query contains invalid SQL that prevents query parsing and identifier extraction. We use recall as the primary measure for schema linking, as it does not penalize generated queries that contain extra identifiers that do not render an answer incorrect in our setting, such as cases when an arbitrary column is referenced in a count function. Charts and tables depicting F1 and precision scores are available in the technical report [26].

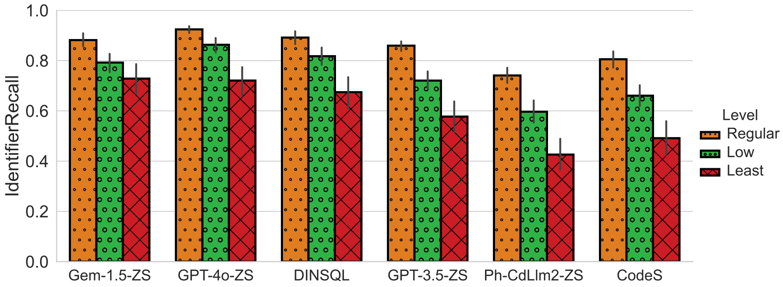

Fig. 9. Native identifier recall scores by model and naturalness level. Error bars set with confidence interval of 0.95. For all models, identifiers in lower naturalness categories yield lower recall scores.

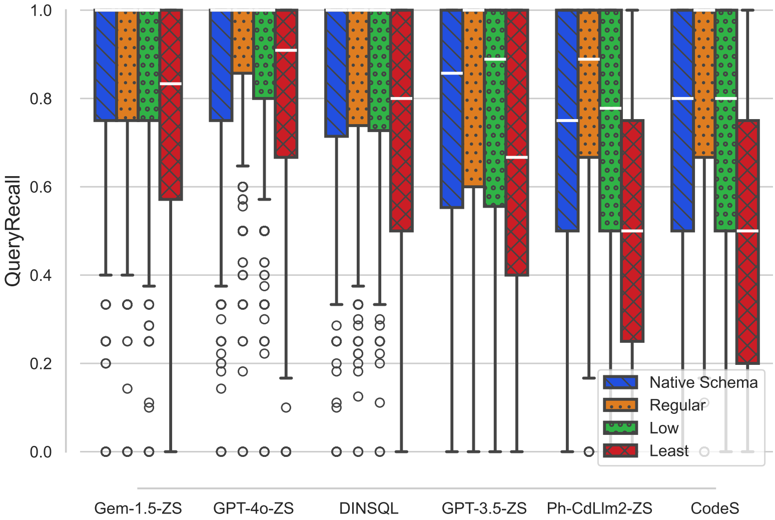

Fig. 10. Schema linking performance across database schema naturalness levels generally yields equal or better performance for higher levels of naturalness, with open source models Phind-CodeLlama2 (Ph-CdLlm2-ZS) and CodeS as well as OpenAI's GPT-3.5 (GPT-3.5-ZS) exhibiting higher sensitivity to changes in naturalness. Zero-shot prompting NL-to-SQL methods are denoted as (ZS).

Identifier-Level Linking Analysis. For an identifier-focused (rather than query-focused) metric, we perform identifier-level linking analysis. We derive recall linking scores for each Native schema identifier I as follows. I_{match} is the count of instances when I is correctly present in a predicted query. I_{gold} is the count of gold queries that contain I.

$$IdentifierRecall = \frac{I_{match}}{I_{gold}} \tag{4}$$

Figure 9 visualizes *IdentifierRecall* of Native identifiers in each naturalness level, and for each LLM. The chart indicates an observable difference in *IdentifierRecall* scores for each naturalness level, with *IdentifierRecall* increasing for higher naturalness levels. These results remain consistent relative to overall model performance across all 5 LLMs and various workflows.

Naturalness Effect on Schema Linking. Overall, we find that schema naturalness has a model-dependent and significant effect on schema linking performance with the highest correlations between *QueryRecall* and query naturalness occurring with the open-source CodeLlama and CodeS models, and the lowest (though still significant) correlations occurring with Google's SoTA Gemini 1.5 Pro and OpenAI's GPT-4o models. The more complex DIN SQL and CodeS workflow *QueryRecall* results are also significantly affected by naturalness level differences.

Both DIN SQL and the CodeS complex NL-to-SQL workflows are sensitive to changes in naturalness, suggesting that these more complex workflows by themselves do not overcome schema naturalness effects. We also see that execution accuracy differences between the GPT-4o zero-shot prompting method and the DINSQL prompt chaining method suggest that applying more complex workflows to high-performing LLMs may be counterproductive for more recent SoTA LLMs.

Figure 10 illustrates *QueryRecall* across schema naturalness levels, and for each LLM. For GPT 3.5, Phind-CodeLlama2, and CodeS, we observe an improvement to *QueryRecall* when converting identifiers in a Native schema to Regular naturalness. This improvement did not manifest for Gemini and GPT-4o when observing the data in aggregate (i.e., between databases) due to their overall high performance relative to the other models, but improvements within databases of lower naturalness are still present (see Figure 11). The recall drop (approximately 20 percent decrease) associated with a modification from both Regular and Low to Least naturalness remains consistent across all LLMs.

Naturalness changes within specific SNAILS database schemas paints a clearer picture of the impact of naturalness. Figure 11 provides a drill-down view of the effect of schema modification on the PILB, SBOD, and NTSB schemas in terms of *QueryRecall*, and for each LLM and schema naturalness level. The center example (PILB) is a highly natural Native schema where schema naturalness modification would not be required. The leftmost example (NTSB) indicates linking performance improvement across all models for a native schema of lower naturalness converted to a higher naturalness schema, and presents a case where naturalness modification will improve NLI performance. The rightmost database (SBOD) represents a Least naturalness schema, and transformation from Native to Regular yields significant improvements for all models. In all cases, we see that reducing naturalness to the Least level consistently degrades *QueryRecall*.

Statistical Significance. Kendall-Tau correlations between the proportion of Least identifiers and *QueryRecall* range from $\tau = -0.16$ (Gemini) to $\tau = -0.28$ (Phind-CodeLlama2), with $P < 0.001$ for all models. Both Regular and Low identifier proportions are significantly correlated with improved outcomes in terms of *QueryRecall.* Identifiers with Regular naturalness show the highest positive Kendall-Tau correlations ranging from $\tau = 0.07$ (Gemini) to $\tau = 0.20$ (Phind-CodeLlama2). Low naturalness identifier proportions correlate positively, but to a lesser degree, with Kendall-Tau values ranging from $\tau = 0.05$ (Phind-CodeLlama2) to $\tau = 0.07$ (Gemini).

Naturalness Effects on Schema Subsetting. We measure the schema subsetting (also known as schema filtering, or table retrieval) in terms of recall, precision, and f1 score, and present the results in Figure 12. We find that for the CodeS finetuned classifier approach, schema naturalness level differences result in observable differences in f1. For the DIN SQL LLM-based approach, naturalness effects are less pronounced, though still present, particularly for Least level schemas.

Performance Over Modified Spider Schemas. Figure 13 shows that with the SNAILS schema renaming artifacts applied to the Spider NL-to-SQL benchmark dev dataset [55], naturalness effects are the most significant between Low and Least levels of naturalness. Performance differences across naturalness levels for the highly natural Spider schemas resemble performance over similarly-natural schemas in the SNAILS collection.

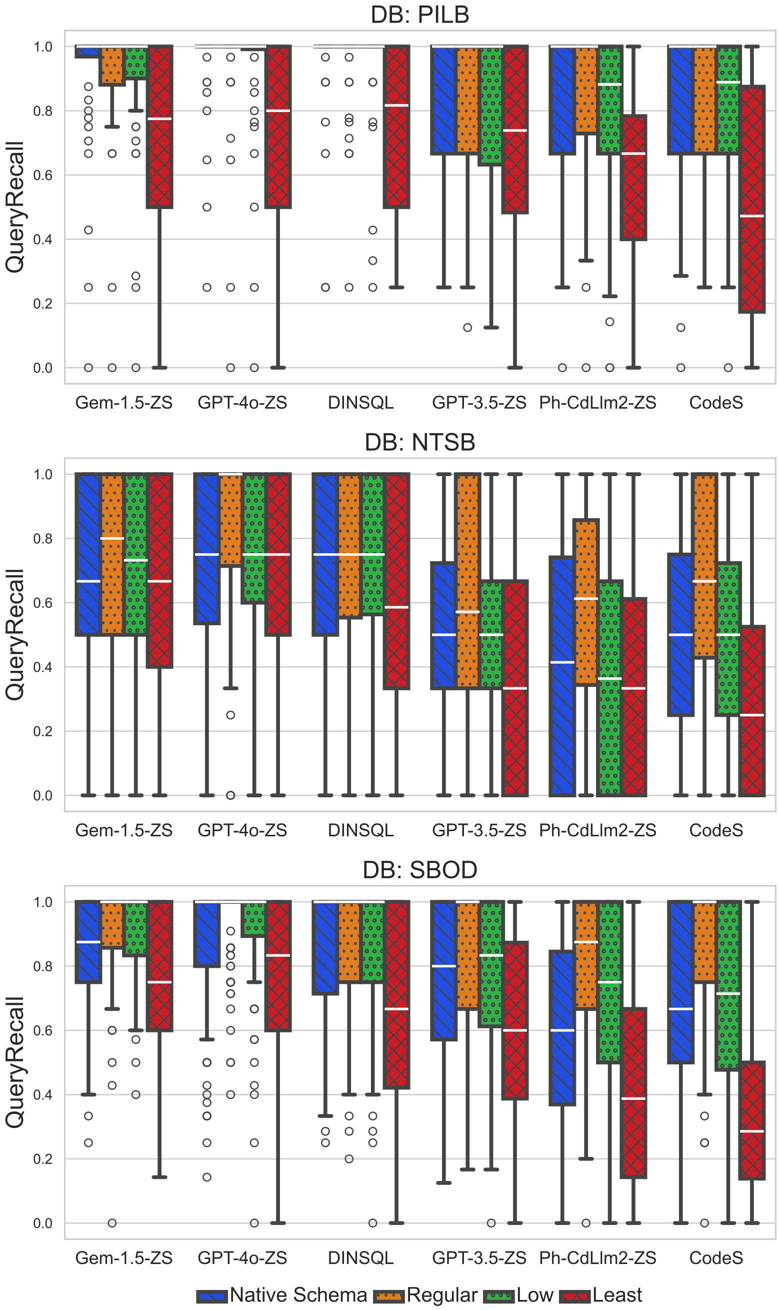

Fig. 11. Schema linking performance (QueryRecall score) changes across 3 example databases' native and virtual schemas. We selected these 3 examples to showcase the diversity of the databases in our collection. PILB Native is a more natural schema with 65 percent Regular, 22 percent Low, and 13 percent Least; NTSB Native contains 42 percent Regular, 34 percent Low, and 24 percent Least; and SBOD Native is the lowest naturalness schema with 24 percent Regular, 49 percent Low, and 27 percent Least.

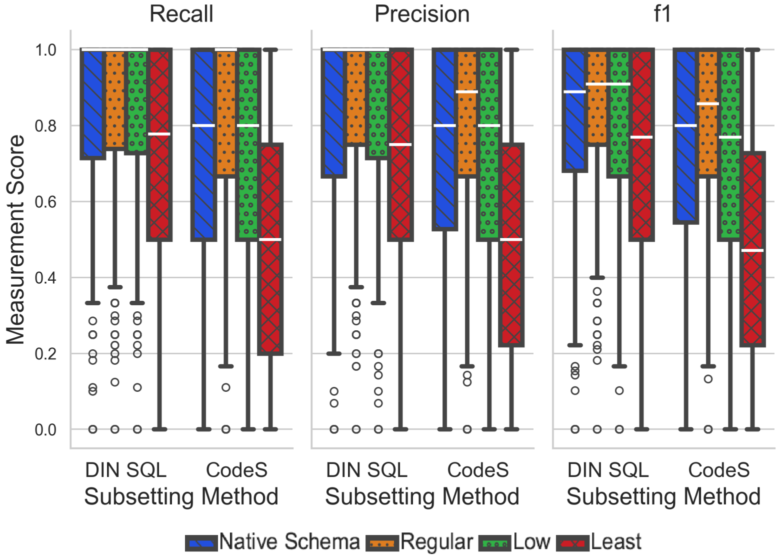

Fig. 12. Schema subsetting performance, measured with recall, precision, and f1 score, varies by naturalness levels for both DIN SQL and CodeS. Measurement Score is Recall, Precision, or f1 respectively.

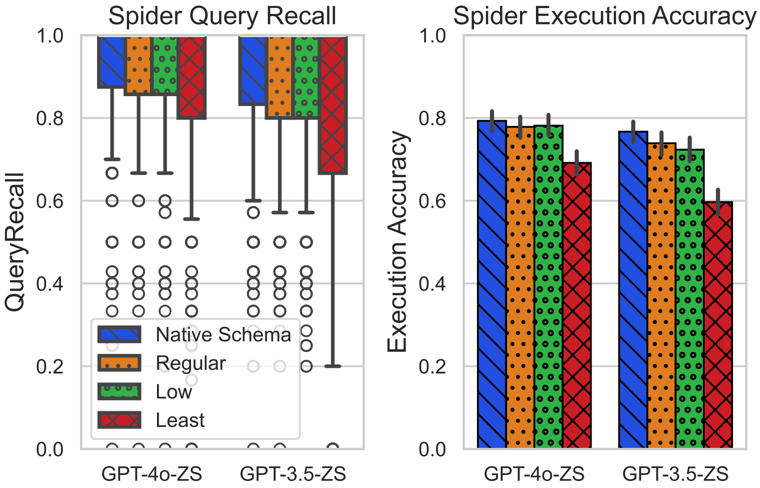

Fig. 13. QueryRecall and Execution Accuracy differences over the Spider [55] dev set modified using SNAILS renaming artifacts.

Additional Charts and Figures. The technical report [26] also provides additional fine-grained results: a more detailed tabular breakdown of execution accuracy by schema and LLM; Precision- and F1-based results; token ratio correlations; and more granular *QueryRecall* correlations and box plots.

6 Discussion and Limitations

The ability to assess the naturalness of existing schemas can inform the feasibility of "hooking up" an NL query interface to an existing database. We believe that practitioners who are considering the integration of an LLM into their database interaction workflows would benefit from making naturalness-focused schema analysis a key step in their integration process.

Other Naming Patterns in Real-World Schemas. To examine naming practices in the real-world, we classified the identifiers of SchemaPile dataset [7] with our CANINE-based classifier, and evaluated the identifiers for other LLM-unfriendly patterns. We observe that whitespace characters within schema identifiers contributes to identifier mutation during inference. That is, rather than encasing a whitespace-containing identifier with brackets or quotes, the LLM hallucinates the identifier into snake or camel case format. We find that in the SchemaPile collection, though whitespace is uncommon (less than 1 percent for both tables and columns), it appears in 808 columns and 63 tables, and is comparable to the proportions in the SNAILS dataset.

Another naming practice that yields disproportionate failures with some LLMs is the presence of the word *table* in the identifier name. In these instances, we find that the LLM tends to drop the word *table* from the name (e.g., table_employee becomes employee). There are over 700 identifiers (less than 1 percent of all identifiers) in the SchemaPile collection that employ this naming pattern.

These observations suggest that although these naming patterns are not necessarily a common occurrence in many real-world schema designs, they do appear in some cases. We suggest that practitioners would benefit from assessing the naming patterns of their database schemas.

Variations in LLM Sensitivity to Naturalness. There are many LLMs to select from for NLIDBs, and we can see even within the select 5 models in our work large variations in NL-to-SQL performance as well as the degree of sensitivity to schema naturalness. The Google Gemini and GPT-4o models demonstrate the highest overall performance, as well as the lowest sensitivity to naturalness differences between Regular and Low levels. Without access to the underlying model architectures and weights, it remains as a black box in our research, and we can merely speculate the reasons why it is not as affected by naturalness as the other 3 models in our study. Generally, we observe that the these models have an overall higher performance, and are less prone to linking errors such as selecting the incorrect identifier from the schema knowledge representation or committing a typo-like hallucination.

Though selecting the most performant model would seem to be an obvious course of action, competing factors such as an organization's policies, budget, or existing vendor contracts, may require the selection of a model that is more sensitive to schema naturalness differences. Thus, we believe that naturalness-aware NLI integration will remain important for at least the practitioners who use LLMs other than Gemini in the set that we have studied.

Modifying Existing Schemas. For already-existing schemas, renaming identifiers is generally a non-trivial effort, particularly for those databases for which documentation has been published and application interfaces have been integrated. Schema modifications may not be necessary (or helpful), if a schema is already classified as highly natural. DBAs should assess current naturalness levels prior to committing to naming modifications. At a minimum, we recommend that any Least identifier be modified to a Regular naturalness level and, if feasible, Low identifiers as well. If renaming a less natural schema's identifiers is not feasible due to integration constraints, we suggest one of two approaches: 1) adopting a naturalness-as-a-view strategy by mapping Native identifiers to Regular naturalness identifiers using SQL views, or 2) a middleware approach that modifies schema knowledge in LLM prompts and generated SQL queries prior to execution on the database. We sketch a rough design of both options in the technical report [26].

We demonstrate a natural schema view proof of concept with our SNAILS database collection and their MS SQL Server instances. For each table and column in the collection's database schemas, we map the Native table or column to its Regular counterpart in the naturalness modified identifier dataset using SQL view creation DDL and a db_nl schema. This enables schema information retrieval for LLM-based NL-to-SQL prompting without prompt or generated query modification while still retaining the underlying Native schema naming patterns required for existing integrations.

In lieu of schema modificaftion, practitioners may elect to employ prompting techniques that augment schema representations with additional metadata or value samples. While these methods may improve schema linking performance in some contexts [29], they greatly increase schema representations on a per-identifier basis. Thus, the cost to do so is high in terms of token efficiency, latency, and implementation complexity, especially for very large schemas.

Designing New Schemas. For new schema development, our results show that making schema identifiers more natural from the start can make databases work better with LLMs. Specifically, database designers should try to avoid Least naturalness identifiers and would likely also benefit from limiting Low naturalness identifiers. Database practitioners can evaluate the naturalness of identifiers using the identifier naturalness classification techniques and model artifacts described in this paper and released publicly by us as part of the SNAILS collection.

Limitations. LLM research is advancing rapidly, and the LLMs represented in this paper may get superseded by newer versions or newer models (e.g., DBRX [50], Arctic [51]). But it does not negate our work's core value–the first in-depth characterization of how schema naturalness affects LLM-based NL-to-SQL–and our new labeled datasets, AI artifacts, and benchmarking framework can be used for future LLMs too. We leave it to future work to also include such very recent LLMs for further benchmark analyses.

We recognize that the correlation statistics indicate a moderate (in some cases only a weak) correlation between naturalness and *IdentifierRecall*. This suggests that other undiscovered factors also influence linking performance; and further research may reveal additional schema- and language-related correlations.

Our selection of 9 database schemas is of course not fully representative of *all* types of schemas available in the real-world. The SNAILS collection will benefit from continued growth in terms of both databases and NL-SQL pairs. We hope our open source datasets and artifacts can be built upon by the database and NLP communities to keep improving LLM-based NL-to-SQL.

Future Work. In addition to extending the SNAILS benchmark artifacts to include additional datasets and artifacts, we identify several NLP+DB directions for future work. First, we wish to ask why and how exactly do different naturalness levels alter schema linking performance so much? Is it due to the tokenization and embedding mechanics? If so, where in the latent space do these altered tokens end up, and how do the encoders make use of them? Second, why do the different foundational LLMs behave so differently? Is it related to their architectures, tokenization, (pre)training data, post-training finetuning process, or some other factors? We believe these open questions have the potential to lead to several interesting new lines of research at the DB and NLP intersection.

7 Related Work

Ontology Mapping. Schema modifications and intermediate representations to enhance performance in a specific context extend beyond NL-to-SQL applications. Mapping relational database schemas to ontologies is an approach used to improve schema-to-schema integration and web application application-database interfaces [54]. This improves the semantic description of underlying data, which is often a desirable feature in web applications that interact within the semantic web [17]. While ontological mapping of a relational database can improve performance in this context; we see less evidence that such an approach is useful or necessary in NL-to-SQL applications, though this may serve as a compelling opportunity for future research.

NL-to-SQL Benchmarks. *Spider* [55], soon to be superseded by a more challenging benchmark for the LLM era, was a popular NL-to-SQL benchmark that still offers a publically-available dataset consisting of 166 multi-table databases and 1,034 NL questions and gold queries over the databases in a development dataset. *Spider-Syn* [12] and Spider-Realistic [12] are extensions of the Spider benchmark that perform NL question synonym replacement to reduce the occurrences of lexical matching between NL question keywords and schema identifiers. *BIRD* [24] is an emergent benchmark containing 95 large databases over 37 domains that seeks to better replicate real-world databases in order to better challenge highly capable LLM-based NL-to-SQL systems. While Spider and its variants as well as BIRD intend to better-replicate real-world database designs, our naturalness-focused analysis indicates that their schema identifiers are more natural than those we encountered in our real-world database selection process (see the statistics in Figure 3). Additionally, Spider and BIRD both evaluate performance using either exact set matching or execution result set comparison while we use the more pragmatic set-superset matching as proposed in [11] and schema linking-specific recall metrics.

Archerfish [11] is a benchmarking framework that relaxes execution matching and accounts for semantic ambiguity in NL questions by allowing for multiple correct answers derived from candidate key analysis. This framework relies on the binary "correct, or not" evaluation approach common to other benchmarks, whereas in addition to relaxed execution matching, SNAILS evaluates target schema linking performance via query identifier recall. Overall, we find that our benchmark and findings complement this existing and ongoing research by enhancing our ability to target specific schema-related aspects of NL-to-SQL performance in future NLI development.

Impacts of Schema on NL-to-SQL Performance. Spider-Syn [12] demonstrates degraded NL-to-SQL performance of language models trained for NL-to-SQL tasks when the occurrence of lexical matching between NL questions and schema identifiers is reduced. This approach differs from our experiments in that it evaluates a LM specifically trained on NL-to-SQL tasks using the Spider training set as opposed to the more general-purpose foundational LLMs evaluated in this work. They also make no apparent attempt to reduce the naturalness of database schema identifiers.

Semantics-preserving schema transformation is a design feature of MT-teql [27], an NL-to-SQL evaluation framework that modifies natural language utterances and schema properties to stress LM robustness. MT-teql provides a holistic view of the effect of NL utterance variances and schema design on LM performance. However, it does not address the question of schema identifier naturalness, nor does it make modifications to schema elements that are necessary for answer generation.

Some recent work has examined the effects of schema ambiguity, where semantically different tables or columns have identical or synonymous names. Schema ambiguity, where a schema contains one or more semantically similar pairs of elements, degrades semantic parsing (i.e., NL-to-SQL) performance by recalling undesired tables or columns in response to a NL question that contains patterns or keywords that align with more than one schema element in the latent space [35]. Documentation, combined with agent-based column selection, can improve Text-to-SQL performance in the presence of data and schema ambiguity [18]. Though we did not focus on ambiguity in our work, identifier naturalness and ambiguity are complementary efforts that provide a potential future direction for the expansion of the SNAILS benchmark artifacts.

Acknowledgments

We thank the members of the UCSD Database Lab and Jingbo Shang for their feedback on this work. This work was supported in part by the U.S. Army Advanced Civil Schooling program and gifts from VMWare. A part of this work used AWS through the CloudBank project, which is supported by NSF grant 1925001. This work was done in part at the U.S. Army Cyber Institute by the first author. OpenAI GPT 3.5 was used to help with LaTeX table and syntax editing.

References

[1] [n. d.]. NPS IRMA Portal. https://irma.nps.gov/Portal/. Accessed: April 2023.

[2] 2022. Report Card Database 2021-22. https://data.nysed.gov/files/essa/21-22/SRC2022.zip. Accessed: May 2023.

[3] Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. LGESQL: Line Graph Enhanced Text-to-SQL Model with Mixed Local and Non-Local Relations. 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)*. Association for Computational Linguistics, Online, 2541–2555. doi:10.18653/v1/2021.acl-long.198

[4] Pankaj Kumar Choudhary. 2022. Naming Conventions in SQL. https://www.c-sharpcorner.com/UploadFile/f0b2ed/what-is-naming-convention/. Last accessed on 2024-01-01.

[5] Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an Efficient Tokenization-Free Encoder for Language Representation. *Transactions of the Association for Computational Linguistics* 10 (2022), 73–91. doi:10.1162/tacl_a_00448

[6] Robert Cook. 2016. Field Data for Assateague Island National Seashore Amphibian and Reptile Inventory. https://irma.nps.gov/DataStore/Reference/Profile/2236826. Accessed: April 2023.

[7] Till Doehmen, Radu Geacu, Madelon Hulsebos, and Sebastian Schelter. 2024. SchemaPile: A Large Collection of Relational Database Schemas. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*.

[8] Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, lu Chen, Jinshu Lin, and Dongfang Lou. 2023. C3: Zero-shot Text-to-SQL with ChatGPT. arXiv:2307.07306 [cs.CL]

[9] Thomas Evans. 2015. Great Smoky Mountains All Taxa Biodiversity Inventory (ATBI) Plot Vegetation Monitoring Database. https://irma.nps.gov/DataStore/Reference/Profile/2221324. Accessed: April 2023.

[10] Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving Text-to-SQL Evaluation Methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 351–360. doi:10.18653/v1/P18-1033

[11] Avrilia Floratou, Fotis Psallidas, Fuheng Zhao, Shaleen Deep, Gunther Hagleither, Wangda Tan, Joyce Cahoon, Rana Alotaibi, Jordan Henkel, Abhik Singla, Alex Van Grootel, Brandon Chow, Kai Deng, Katherine Lin, Marcos Campos, Venkatesh Emani, Vivek Pandit, Victor Shnayder, Wenjing Wang, and Carlo Curino. 2024. NL2SQL is a solved problem... Not!. In *Proceedings of the CIDRDB 2024 Conference*. https://www.cidrdb.org/cidr2024/papers/p74-floratou.pdf

[12] Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. 2021. Towards Robustness of Text-to-SQL Models against Synonym Substitution. 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)*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, Online, 2505–2515. doi:10.18653/v1/2021.acl-long.195

[13] 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. 14 pages. doi:10.14778/3641204.3641221

[14] General Services Administration. 2023. Security Policy for Generative Artificial Intelligence (AI) Large Language Models (LLMs). https://www.gsa.gov/directives-library/security-policy-for-generative-artificial-intelligence-ai-large-language-models-llms. Last accessed on 2024-05-28.

[15] Frederic Piesschaert Gert Van Spaendonk, Jo Loos. [n. d.]. Database naming conventions. https://inbo.github.io/tutorials/tutorials/database_conventions/. Last accessed on 2024-01-01.

[16] Sree Hari Krishnan Parthasarathi, Lu Zeng, and Dilek Hakkani-Tür. 2023. Conversational Text-to-SQL: An Odyssey into State-of-the-Art and Challenges Ahead. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1–5. doi:10.1109/ICASSP49357.2023.10096170

[17] Mohamed A. G. Hazber, Ruixuan Li, Yuxi Zhang, and Guandong Xu. 2015. An Approach for Mapping Relational Database into Ontology. In *2015 12th Web Information System and Application Conference (WISA)*. 120–125. doi:10.1109/WISA.2015.25

[18] Zezhou Huang, Pavan Kalyan Damalapati, and Eugene Wu. 2023. Data Ambiguity Strikes Back: How Documentation Improves GPT's Text-to-SQL. In *NeurIPS 2023 Second Table Representation Learning Workshop*. https://openreview.net/

forum?id=FflKTuIRTD

[19] Klamath Inventory and Monitoring Network. 2021. Exotic and Invasive Plants Monitoring Database. https://irma.nps.gov/DataStore/Reference/Profile/2288667. Accessed: April 2023.

[20] Seth Judge and Kevin Kozar. 2023. Pacific Island Network Landbird Monitoring Dataset. https://irma.nps.gov/DataStore/Reference/Profile/2300107. doi:10.57830/2300107 Accessed: April 2023.

[21] Tobias Kuhn. 2014. A Survey and Classification of Controlled Natural Languages. *Computational Linguistics* 40, 1 (03 2014), 121–170. doi:10.1162/COLI_a_00168 arXiv:https://direct.mit.edu/coli/article-pdf/40/1/121/1812691/coli_a_00168.pdf

[22] Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023. RESDSQL: decoupling schema linking and skeleton parsing for text-to-SQL. 9 pages. doi:10.1609/aaai.v37i11.26535

[23] Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024. CodeS: Towards Building Open-source Language Models for Text-to-SQL. *Proc. ACM Manag. Data* 2, 3, Article 127 (May 2024), 28 pages. doi:10.1145/3654930

[24] 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. 2024. Can LLM already serve as a database interface? a big bench for large-scale database grounded text-to-SQLs. 28 pages.

[25] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023. StarCoder: may the source be with you! (2023). arXiv:2305.06161 [cs.CL]

[26] Kyle Luoma and Arun Kumar. 2024. *SNAILS: Schema Naturalness Assessments for Improved LLM Systems.* Technical Report. La Jolla, CA, USA.

[27] Pingchuan Ma and Shuai Wang. 2021. MT-Teql: Evaluating and Augmenting Neural NLIDB on Real-World Linguistic and Schema Variations. *Proc. VLDB Endow.* 15, 3 (nov 2021), 569–582. doi:10.14778/3494124.3494139

[28] Ian Muirhead. 2021. Northern Great Plains Fire Management: FFI Database. https://irma.nps.gov/DataStore/Reference/Profile/2297267. Accessed: April 2023.

[29] Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. 2023. Enhancing Few-shot Text-to-SQL Capabilities of Large Language Models: A Study on Prompt Design Strategies. arXiv:2305.12586 [cs.CL] https://arxiv.org/abs/2305.12586

[30] National Center for Statistics and Analysis. 2022. Overview of the 2021 Crash Investigation Sampling System. https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813397 Traffic Safety Facts Research Note. Report No. DOT HS 813 397.

[31] OpenAI. [2022]. ChatGPT: Optimizing Language Models for Dialogue. https://openai.com/blog/chatgpt/.

[32] OpenAI. 2023. OpenAI API Documentation. https://platform.openai.com/docs/guides/gpt. Last accessed on 2023-10-30.

[33] Oracle. 2024. Database Object Names and Qualifiers. https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Database-Object-Names-and-Qualifiers.html. Last accessed on 2025-01-04.

[34] Oracle. 2024. Table Naming Standards and Conventions. https://docs.oracle.com/cd/E92917_01/PDF/8.1.x.x/common/HTML/DM_Naming/2_Table_and_Column_Naming_Standards.htm. Last accessed on 2025-01-04.

[35] Simone Papicchio, Paolo Papotti, and Luca Cagliero. 2024. Evaluating Ambiguous Questions in Semantic Parsing. In *2024 IEEE 40th International Conference on Data Engineering Workshops (ICDEW).* 338–342. doi:10.1109/ICDEW61823.2024.00050

[36] Simone Papicchio, Paolo Papotti, and Luca Cagliero. 2024. QATCH: benchmarking SQL-centric tasks with table representation learning models on your data. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23).* Curran Associates Inc., Red Hook, NY, USA, Article 1348, 20 pages.

[37] Terence Parr, Sam Harwell, and Kathleen Fisher. 2014. Adaptive LL(*) Parsing: The Power of Dynamic Analysis. In *Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages and Applications* (Portland, Oregon, USA) *(OOPSLA '14).* Association for Computing Machinery, New York, NY, USA, 579–598. doi:10.1145/2660193.2660202

[38] Phind. 2023. Phind-CodeLlama-34B-v2. https://huggingface.co/Phind/Phind-CodeLlama-34B-v2.

[39] Marie-Laurence Poujois. 2021. Localized Demo Databases Now Available for SAP Business One 10.0 FP 2011. https://blogs.sap.com/2021/01/29/localized-demo-databases-now-available-for-sap-business-one-10.0-fp-2011/. Accessed: April 2023.

[40] Mohammadreza Pourreza and Davood Rafiei. 2024. DIN-SQL: decomposed in-context learning of text-to-SQL with self-correction. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 1577, 10 pages.

[41] G. A. Radja, E.-Y. Noh, and F. Zhang. 2022. Crash Investigation Sampling System 2021 analytical user's manual. Accessed: April 2023.

[42] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code Llama: Open Foundation Models for Code. arXiv:2308.12950 [cs.CL]

[43] Tushar Sharma, Marios Fragkoulis, Stamatia Rizou, Magiel Bruntink, and Diomidis Spinellis. 2018. Smelly Relations: Measuring and Understanding Database Schema Quality. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice* (Gothenburg, Sweden) *(ICSE-SEIP '18)*. Association for Computing Machinery, New York, NY, USA, 55–64. doi:10.1145/3183519.3183529

[44] StackOverflow. [n. d.]. Database, Table and Column Naming Conventions? https://stackoverflow.com/questions/7662/database-table-and-column-naming-conventions. Last accessed on 2024-01-01.

[45] Charles Stefanic. 2021. Wildlife Observations Database: Craters of the Moon National Monument and Preserve 1921-2021. https://irma.nps.gov/DataStore/Reference/Profile/2192964. Accessed: April 2023.

[46] Alane Laughlin Suhr, Kenton Lee, Ming-Wei Chang, and Pete Shaw. 2020. Exploring Unexplored Generalization Challenges for Cross-Database Semantic Parsing. In *ACL 2020*.

[47] Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv:2403.05530 [cs.CL]

[48] Gemini Team. 2024. Gemini: A Family of Highly Capable Multimodal Models. arXiv:2312.11805 [cs.CL]

[49] PaLM 2 Team. 2023. *PaLM 2 Technical Report.* Technical Report. arXiv:2305.10403 [cs.CL]

[50] The Mosaic Research Team. 2024. Introducing DBRX: A New State-of-the-Art Open LLM. https://www.databricks.com/blog/introducing-dbrx-new-state-art-open-llm.

[51] The Snowflake Research Team. 2024. Snowflake Arctic: The Best LLM for Enterprise AI - Efficiently Intelligent, Truly Open. https://www.snowflake.com/blog/arctic-open-efficient-foundation-language-models-snowflake/.

[52] Bailin Wang, Richard Shin, Xiaodong Liu, Alex Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *ACL 2020*. https://www.microsoft.com/en-us/research/publication/rat-sql-relation-aware-schema-encoding-and-linking-for-text-to-sql-parsers/

[53] Lihan Wang, Bowen Qin, Binyuan Hui, Bowen Li, Min Yang, Bailin Wang, Binhua Li, Jian Sun, Fei Huang, Luo Si, and Yongbin Li. 2022. Proton: Probing Schema Linking Information from Pre-Trained Language Models for Text-to-SQL Parsing. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 1889–1898. doi:10.1145/3534678.3539305

[54] Zhuoming Xu, Shichao Zhang, and Yisheng Dong. 2006. Mapping between Relational Database Schema and OWL Ontology for Deep Annotation. In *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*. 548–552. doi:10.1109/WI.2006.114

[55] 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*. Association for Computational Linguistics, Brussels, Belgium.

[56] Lu Zeng, Sree Hari Krishnan Parthasarathi, and Dilek Hakkani-Tur. 2023. N-Best Hypotheses Reranking for Text-to-SQL Systems. In *2022 IEEE Spoken Language Technology Workshop (SLT)*. 663–670. doi:10.1109/SLT54892.2023.10023434

[57] Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Shen Wang, Huzefa Rangwala, and George Karypis. 2023. NameGuess: Column Name Expansion for Tabular Data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 13276–13290. doi:10.18653/v1/2023.emnlp-main.820

Received July 2024; revised September 2024; accepted November 2024