International Journal of Geo-Information

MDPI

Article

GPT-Based Text-to-SQL for Spatial Databases

Hui Wang [1,*,†], Li Guo [2,†], Yubin Liang [1], Le Liu [3] and Jiajin Huang [4,5]

[1] Department of Geography, Tianjin Normal University, Tianjin 300387, China; ybliang@tjnu.edu.cn
[2] Institute of Geospatial Information, Information Engineering University, Zhengzhou 450001, China; gl_750312@163.com
[3] Faculty of Architectural Engineering, Tianjin University, Tianjin 300350, China; liule0328@tju.edu.cn
[4] Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China; jhuang@bjut.edu.cn
[5] Beijing International Collaboration Base on Brain Informatics and Wisdom Services, Beijing 100124, China
* Correspondence: wanghui@tjnu.edu.cn
† These authors contributed equally to this work.

Abstract

Text-to-SQL for spatial databases enables the translation of natural language questions into corresponding SQL queries, allowing non-experts to easily access spatial data, which has gained increasing attention from researchers. Previous research has primarily focused on rule-based methods. However, these methods have limitations when dealing with complicated or unknown natural language questions. While advanced machine learning models can be trained, they typically require large labeled training datasets, which are severely lacking for spatial databases. Recently, Generative Pre-Trained Transformer (GPT) models have emerged as a promising paradigm for Text-to-SQL tasks in relational databases, driven by carefully designed prompts. In response to the severe lack of datasets for spatial databases, we have created a publicly available dataset that supports both English and Chinese. Furthermore, we propose a GPT-based method to construct prompts for spatial databases, which incorporates geographic and spatial database knowledge into the prompts and requires only a small number of training samples, such as 1, 3, or 5 examples. Extensive experiments demonstrate that incorporating geographic and spatial database knowledge into prompts improves the accuracy of Text-to-SQL tasks for spatial databases. Our proposed method can help non-experts access spatial databases more easily and conveniently.

Keywords: spatial databases; Text-to-SQL; GPT

Academic Editors: Wolfgang Kainz and Eliseo Clementini

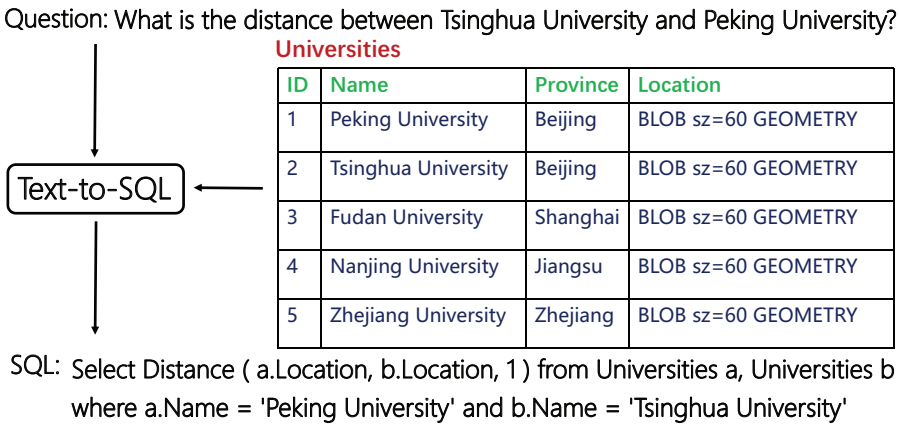
Received: 20 May 2025
Revised: 18 July 2025
Accepted: 22 July 2025
Published: 24 July 2025

Citation: Wang, H.; Guo, L.; Liang, Y.; Liu, L.; Huang, J. GPT-Based Text-to-SQL for Spatial Databases. *ISPRS Int. J. Geo-Inf.* **2025**, *14*, 288. https://doi.org/10.3390/ijgi14080288

1. Introduction

According to a study by [1], 80% of big data is spatial data. Spatial databases, which offer highly specialized spatial query and analysis capabilities, have thus become essential tools for managing this data. While experts can interact with these databases using SQL, the language can be challenging for non-experts, limiting their ability to fully leverage this data managed by spatial databases.

Text-to-SQL for spatial databases offers a promising solution by translating the natural language question/text (NLQ/Text) into the corresponding SQL query, as depicted in Figure 1. This enables non-experts to easily access spatial data and has garnered increasing attention from researchers.



ID	Name	Province	Location
1	Peking University	Beijing	BLOB sz=60 GEOMETRY
2	Tsinghua University	Beijing	BLOB sz=60 GEOMETRY
3	Fudan University	Shanghai	BLOB sz=60 GEOMETRY
4	Nanjing University	Jiangsu	BLOB sz=60 GEOMETRY
5	Zhejiang University	Zhejiang	BLOB sz=60 GEOMETRY

Figure 1. An example of Text-to-SQL for spatial databases.

Previous studies have investigated the problem of translating natural language questions into SQL queries for spatial databases [2–4]. These studies can be summarized as rule-based methods. Although these methods do not require large amounts of training data, they have limitations when dealing with complicated or unknown natural language questions.

These limitations could theoretically be addressed by training a machine learning model. However, the model requires a labeled training dataset that not only covers a diverse range of spatial natural language questions but also includes the corresponding correct SQL queries. Additionally, the dataset must be sufficiently large to ensure that the trained machine learning model achieves satisfactory performance. Unfortunately, due to the complexity of spatial queries, creating such a dataset can be both time-consuming and labor-intensive.

Recently, Generative Pre-Trained Transformer (GPT) models, such as GPT-4 and DeepSeek, have demonstrated impressive performance across a wide range of tasks. These models are designed to take a text-based input, known as a prompt, and generate human-like responses. When using a GPT model to generate an accurate SQL query, one critical factor to consider is the construction of the prompt. The prompt should be based on the natural language question and the corresponding database schema. Studies [5–7] indicate that including examples related to the natural language question in the prompt enables GPT models to generate outputs that better align with user expectations.

In this paper, we propose a GPT-based method to construct prompts for spatial databases, which incorporates geographic and spatial database knowledge into the prompts and requires only a small number of training samples, such as 1, 3, or 5 examples. Additionally, we have created a dataset containing spatial natural language questions paired with their corresponding SQL queries, available in both English and Chinese. This dataset can be used for model training and performance evaluation. Furthermore, our framework requires only a small number of training samples, such as 1, 3, or 5 examples, to effectively guide the GPT model in generating accurate SQL queries.

The remainder of this paper is organized as follows. Section 2 reviews related work on Text-to-SQL tasks for spatial databases. Section 3 presents our proposed method. Section 4 outlines the experimental design used to evaluate our method. Section 5 presents the results of the experiments. Section 6 discusses the limitations of our method. Finally, Section 7 concludes the paper.

2. Related Work

High-quality datasets are essential for studying and evaluating the Text-to-SQL task. Researchers have developed several annotated datasets for Text-to-SQL, including GeoQuery [8], Restaurants [9], Scholar [10], WikiSQL [11], Spider [12], CSpider [13], DuSQL [14],

Spider-DK [15], Spider-Syn [16], and Spider-SSP [17]. These datasets consist of <NLQ/Text, SQL > pairs and are widely utilized in academic research on Text-to-SQL.

To facilitate the movement of Text-to-SQL from academic research to real applications, Alibaba Group has partnered with institutions such as the University of Hong Kong to introduce the BIRD dataset [18]. This dataset focuses on extensive and authentic database content, serving as a large-scale benchmark for databases.

These datasets mentioned above all focus on relational databases. Although the GeoQuery and Restaurants datasets are related to the geographic domain, they still lack the relevant data for spatial queries. Consequently, there is still a significant gap in Text-to-SQL datasets specifically designed for spatial databases.

The advancement of deep learning, along with the availability of large-scale Text-to-SQL datasets, has significantly accelerated research on Text-to-SQL for relational databases. Several methods, including Seq2Seq [19], IRNet [20], RYANSQL [21], and SQLova [22], along with pre-trained models such as TaBERT [23], TaPas [24], and Grappa [25], have been proposed and have demonstrated promising results.

Among these methods, SpatialNLI [26,27] primarily focuses on addressing the semantic ambiguity of spatial entities, such as determining whether 'Mississippi' in a natural language question refers to a river or a state. However, the training datasets used by SpatialNLI are still designed for relational databases rather than spatial databases.

Due to the lack of the data of spatial queries in the existing Text-to-SQL datasets, models trained by current Text-to-SQL datasets with deep learning methods often face challenges when applied directly to the Text-to-SQL task for spatial databases. As a result, current research on Text-to-SQL for spatial databases primarily employs rule-based methods [2–4]. This is largely because rule-based methods do not require large-scale training datasets. However, while rule-based methods are suitable for simple user queries, it is challenging, if not impossible, to design SQL templates in advance for various scenarios or domains.

Recently, GPT models have emerged as a new paradigm for the Text-to-SQL task [28–30]. Different from prior studies, the core challenge in GPT-based Text-to-SQL solutions lies in how to guide the model to generate correct SQL queries, specifically how to construct an effective prompt. Many studies have proposed various approaches for prompt construction [4,7,31–33]. However, these approaches are primarily focused on relational databases and do not account for the unique complexities of spatial databases. As a result, directly applying them to the Text-to-SQL task for spatial databases does not yield satisfactory results, as we will demonstrate in our experiments.

The growing demand for spatiotemporal data and modeling in geosciences has greatly advanced geospatial code generation technology. Refs. [34–38] specifically examine the use of large language models for generating geospatial analysis code. For geospatial SQL generation, Ref. [39] briefly reports on the potential of using GPT models for Text-to-SQL tasks targeting spatial databases.

Despite these advancements, methods for constructing prompts to address Text-to-SQL tasks for spatial databases remain limited. In this study, we aim to develop a method of prompt construction for the Text-to-SQL task specifically designed for spatial databases, while simultaneously creating a dedicated dataset for training and evaluating this task.

3. Method

3.1. Overall Framework

Our method consists of two steps: (1) We send the prompt to the GPT model, which generates an SQL query based on the provided prompt. (2) The generated SQL query is then executed in the spatial database management system (SDBMS). If no exceptions occur, the SQL query is considered the final prediction from the GPT model. If an exception is detected, a schema tip (e.g., "the table 't' does not exist in the database and should not be used") is extracted from the exception feedback and incorporated back into the prompt, after which the process is repeated. The procedure terminates once a predefined repetition threshold is reached.

The prompt in our method consists of six components: (1) SDBMS knowledge, (2) examples of natural language questions similar to the target question along with their corresponding SQL queries, (3) the database schema, (4) a geographic description of each table in the database, (5) schema tips, and (6) the target question.

Figure 2 illustrates the overall framework of our method, while Figure 3 provides a snippet of the prompt. The details of the six components of the prompt are outlined in the following subsections.

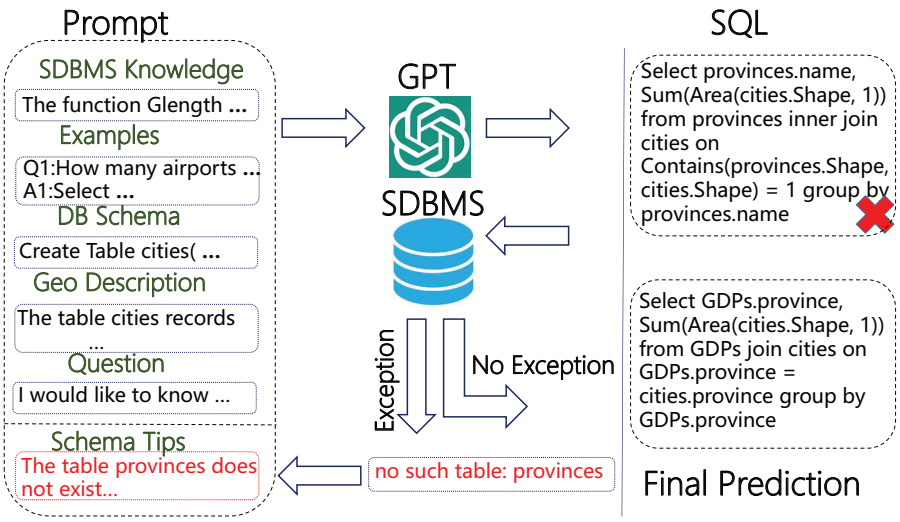


Figure 2. Overall framework of our method.

3.2. SDBMS Knowledge

SDBMSs are typically implemented by means of spatial extensions to relational databases and are offered in several types, such as PostgreSQL with PostGIS, Oracle Spatial, SQL Server Spatial, and SQLite with SpatiaLite. These SDBMSs provide spatial functions to support various spatial queries and analysis capabilities. However, the names of spatial functions corresponding to identical capabilities can differ across various SDBMSs. For instance, when calculating the area of a geometric object, the function is named STArea in SQL Server, whereas in SpatiaLite, the corresponding function is simply designated as Area. Furthermore, within a single SDBMS, the names of these functions may vary across different versions.

These inconsistencies may cause a mismatch between the function names in the SQL query predicted by the GPT model and those provided by the SDBMS used.

Additionally, within the same SDBMS, some spatial functions used for calculations may have different parameterizations, which can result in differing outcomes. For example, in SpatiaLite, the results of Distance (geom1 Geometry, geom2 Geometry, 1) and Distance (geom1 Geometry, geom2 Geometry) differ, with the former providing a more precise result.

```
You are an expert in the SpatiLite spatial database and highly skilled in writing SQL queries.
In SpatiLite,
The function Area(s Surface, 1) returns the area of s, measured in meters.
The function GLength(c Curve, 1) returns the length of c, measured in meters.
...

/* Some SQL examples are provided based on similar problems: */
/* Answer the following: I would like to know the total area of the Beijing-Tianjin-Hebei region. Select Sum(Area(provinces.Shape, 1)) from provinces where provinces.name = '北京市' or provinces.= '天津市' or provinces.name = '河北省'

/* Answer the following: Which provinces border Guangdong Province, and what is the area of province? */
Select b.name , Area(b.Shape, 1) from provinces a inner join provinces b On Touches(a.Shape, b.Shape) 1 where a.name = '广东省'
...
/* Given the following database schema: */
CREATE TABLE 'airports' (
IATA TEXT PRIMARY KEY,
runway_number INTEGER,
airfield_area_class TEXT,
Name TEXT,
Location POINT)

The 'airports' table records information about various airports in China.

CREATE TABLE 'cities' (
name TEXT PRIMARY KEY,
administrative_division_code TEXT,
province TEXT REFERENCES "GDPs" (province),
Shape MULTIPOLYGON)

The 'cities' table records information about various cities in China.

/* Answer the following with no explanation: What is the total area of the three northeastern provinces? The three northeastern provinces refer to Liaoning, Jilin, and Heilongjiang. Liaoning is represented name '辽宁省', Jilin is represented by the name '吉林省', and Heilongjiang is represented by the name '黑龙江省'. The table "provinces" does not exist in the database and should not be used. */
SELECT
```

Figure 3. A snippet of the prompt from our method.

To ensure that the GPT model accurately utilizes the functions provided by the SDBMS used and meets the user' specific computational requirements, we first specify the SDBMS used in the prompt. For example, "You are an expert in the SpatiaLite spatial database and highly skilled in writing SQL queries". In our experiments, we use SpatiaLite.

Additionally, we provide clear explanations of some commonly used functions in the prompt, such as GLength, Area, Distance, Centroid, Touches, and Intersects. For example, "The function GLength (c Curve, 1) returns the length of curve c, measured in meters".

3.3. Examples

We use the Masked Question Similarity Selection method [6,33] to measure the similarity between the target question and questions in the training dataset. Next, we select the K most similar questions, along with their corresponding SQL queries, from the training dataset as examples.

3.4. Database Schema

The database schema defines the structure of the database, including tables, columns, relationships between tables, data types, and additional elements. In this section, we use a Code Representation [40,41] to describe the corresponding database schema. Specifically, as shown in Figure 3, it directly presents the "CREATE TABLE" SQL statements.

3.5. Geographic Descriptions

In Text-to-SQL tasks, adding description for each table in the prompt may help clarify table semantics and reduce ambiguity. In this section, we add a geographic region description for each table in the prompt. This is based on our observation that, in spatial databases, the data in each table typically corresponds to a specific geographic region. For example, in the Traffic database we use, the 'roads' table contains information about roads in Nanjing. There is a category of questions, such as "What is the total length of each road in Nanjing?" and "What is the total length of each road?", which we refer to as questions related to regions. These two questions are essentially equivalent in meaning. However, when faced with the question "What is the total length of each road in Nanjing?", the GPT model often misinterprets 'Nanjing' as a column value in the database, leading to the generation of an incorrect SQL query.

To address this issue, we explicitly describe the geographic region associated with each table in the prompt. For instance, for the 'roads' table, we include the description "The 'roads' table records information about roads in Nanjing.".

3.6. Question

In this section, we adopt a simple question representation in DAIL [33], where the natural language question is prefixed with "Answer the following with no explanation:" and is followed by "SELECT ". When generating SQL queries, the GPT model sometimes provides explanations for certain terms. The phrase "with no explanation" helps ensure that GPT only returns the answer.

3.7. Schema Tips

In Text-to-SQL tasks, self-correction with execution feedback may help improve prediction accuracy. In this section, we focus on handling three types of feedback: nonexistent table names, nonexistent column names, and ambiguous column names. This is because foundational data (e.g., administrative divisions or roads) often appears across multiple spatial databases. Consequently, table or column names in the examples may share identical or similar names or meanings with those in the target question. As shown in Figure 3, the term 'provinces' appears in both the examples and the target question. Such overlap can lead the GPT model to generate SQL queries referencing tables or columns from the examples that do not exist in the given database schema.

To enable the GPT model to correctly reference tables or columns based on the given database schema, we extract three types of schema tips from the exception feedback in Step (2) and incorporate them into the prompt.

- When the exception feedback is "no such table: t", we extract the table name 't' and append the following sentence, "The table 't' does not exist in the database and should not be used.", to the prompt.
- When the exception feedback is "no such column: c", we extract the column name 'c' and its corresponding table name 't', and append the following sentence, "The column 'c' does not exist in the table 't' and should not be used.", to the prompt.
- When the exception feedback is "ambiguous column name: c", we append the following sentence, "Columns in multiple tables may have the same name, so ambiguity should be avoided.", to the prompt.

4. Experiment

4.1. Experiment Dataset

Due to the limited availability of datasets for spatial databases in Text-to-SQL tasks, we have developed a dedicated dataset to evaluate our proposed method. Datasets for relational databases, such as BIRD [18] and Spider [12], which are among the most widely used, are typically built using SQLite, the world's most popular database engine. To maintain consistency, our dataset is built on SpatiaLite, which extends SQLite with comprehensive Spatial SQL capabilities.

The developed dataset comprises four types of spatial databases: Ada (Administrative Divisions), Edu (Education), Tourism, and Traffic. All data is sourced from real-world information in China.

In some datasets for relational databases, such as BIRD [18], a Text-to-SQL pair consists of both a text and an SQL query. The text not only includes the question but also contains evidence, which refers to supplementary knowledge typically provided by experts. To ensure that our dataset supports both Chinese and English questions, we have added a 'name' field to each Text-to-SQL pair, which specifies the actual values of certain phrases in the database.

For example, in the question "What is the total area of the three northeastern provinces? The three northeastern provinces refer to Liaoning, Jilin, and Heilongjiang.", the phrase 'Liaoning' corresponds to '辽宁省' (in Chinese) in the database, 'Jilin' corresponds to '吉林省' (in Chinese), and 'Heilongjiang' corresponds to '黑龙江省' (in Chinese). Hence, in the 'name' field, we include the description "Liaoning is represented by the name '辽宁省', Jilin is represented by the name '吉林省', and Heilongjiang is represented by the name '黑龙江省'".

Additionally, we refer to the procedure of CSpider [13], a Chinese dataset translated from the English dataset Spider [12], to provide Chinese equivalents for the fields of question, evidence, and name. The details are outlined in Table 1.

In spatial databases, certain natural language questions can be performed using SQL with either non-spatial or spatial columns. For instance, the 'rivers' table in the Ada database contains nine columns, including a non-spatial column 'length' and a spatial column 'shape', the latter being of 'geometry' type. To query the length of a river, you can directly access the 'length' column or calculate it using the spatial function 'GLength (shape, 1)'. Similarly, the 'cities' table in the Edu database contains 10 columns, including a non-spatial column 'province' and a spatial column 'shape', while the 'provinces' table in the Edu database contains four columns, including a spatial column 'shape'. To determine the province of a city, you can query the 'province' column in the 'cities' table, or use the

spatial function 'Within (cities.shape, provinces.shape)' to determine if the city lies within a specific province.

Table 1. A Text-to-SQL pair from our developed dataset, where the fields 'questionCHI', 'evidenceCHI', and 'nameCHI' represent the Chinese equivalents of the fields 'question', 'evidence', and 'name', respectively.

question: What is the total area of the three northeastern provinces?
questionCHI: 东三省的总面积是多少?
evidence: The three northeastern provinces refer to Liaoning, Jilin, and Heilongjiang.
evidenceCHI: 东三省指辽宁省, 吉林省, 黑龙江省.
name: Liaoning is represented by the name '辽宁省', Jilin is represented by the name '吉林省', and Heilongjiang is represented by the name '黑龙江省'.
nameCHI: 辽宁省以'辽宁省' 为名称表示, 吉林省以'吉林省' 为名称表示, 黑龙江省以'黑龙江省' 为名称表示.
SQL: Select Sum (Area (Shape, 1)) from cities where province in ('辽宁省', '吉林省', '黑龙江省')

In practical applications, we can calculate the similarity between phrases in the 'SQL' field and the corresponding column values in the database, selecting the value with the highest similarity as the actual value in the 'SQL' field.

We call these non-spatial columns, such as 'length' and 'province', whose values can be inferred or calculated from spatial columns or functions, as derived columns. To enhance our dataset and fully evaluate the performance of our method, we transform the Ada, Edu, and Traffic databases by retaining the derived columns in one version and removing them in another. This process results in two versions of each database: Ada^1, Edu^1, Traffic^1 (with derived columns) and Ada^2, Edu^2, Traffic^2 (without derived columns).

We refer to the spatial questions in [2,3] (each containing 100 questions, with the spatial databases and corresponding SQL queries not publicly available), as well as the general questions in BIRD and Spider, to design a set of 200 questions. Within each database type (Ada^1 and Ada^2, Edu^1 and Edu^2, Traffic^1 and Traffic^2), the questions are identical. In the Ada^1, Edu^1, and Traffic^1 databases, due to the presence of derived columns, a spatial question can be addressed by multiple SQL queries, using either non-spatial columns (general SQL queries) or spatial columns (spatial SQL queries). In contrast, in the Ada^2, Edu^2, and Traffic^2 databases, where derived columns have been removed, spatial questions require spatial columns (i.e., spatial SQL queries). To enhance our dataset, we provide multiple SQL queries for each of the 200 questions whenever possible, resulting in a total of 296 unique SQL queries. The details of the developed dataset are described in Table 2.

Table 2. Detailed statistics for our developed dataset.

Name	Tables	Columns	PKs	FKs	Qs	Rs	SQLs
Ada^1	6	53	6	1	56	7	90
Ada^2	6	49	6	0	56	7	56
Edu^1	3	26	3	3	44	0	93
Edu^2	3	22	3	0	44	0	44
Tourism	6	57	6	2	33	1	33
Traffic^1	11	55	11	4	67	26	81
Traffic^2	11	51	11	3	67	26	67

In the table, the horizontal axis indicates the number of tables, columns, primary keys (PKs), foreign keys (FKs), questions (Qs), questions related to geographic regions (Rs, such as the one "What is the total length of each road in Nanjing?" in Section 3.5), and SQL queries in each database.

4.2. Experiment Methods

We conduct an experimental evaluation using the following methods based on the GPT model:

- SSpa: We have named our method SSpa, which consists of six components. Among them, the examples, database schema, and question are common components of GPT-based methods for relational databases in Text-to-SQL tasks, while the SDBMS knowledge, geographic descriptions, and schema tips represent the innovative aspects of our method.
- DAIL-SQL: DAIL-SQL, developed by Alibaba Group, is a highly effective and efficient method designed to optimize the application of GPT models in Text-to-SQL tasks. Notably, equipped with GPT-4, DAIL-SQL achieved first place on the Spider leaderboard with an execution accuracy of 86.6%, setting a new benchmark. DAIL-SQL is an excellent representative of GPT-based methods for relational databases in Text-to-SQL tasks, consisting mainly of examples, database schema, and question. Hence, we use DAIL-SQL as our baseline method.
- SSpa-SDBMS: Compared to the SSpa method, SSpa-SDBMS removes the SDBMS knowledge component while retaining the geographic descriptions and schema tips.
- SSpa-Geo: Compared to the SSpa method, SSpa-Geo removes the geographic descriptions component while retaining the SDBMS knowledge and schema tips.
- SSpa-Tips: Compared to the SSpa method, SSpa-Tips removes the schema tips component while retaining the SDBMS knowledge and geographic descriptions.

To ensure a fair comparison, we employ GPT-4 for all methods evaluated in this study, given that DAIL-SQL achieves the best performance with GPT-4. To enhance the robustness and reliability of our method, we conduct supplementary comparative experiments using DeepSeek. All methods are accessed through the OpenAI API.

To eliminate any influence from randomness, we set the temperature parameter to 0 by default for both GPT-4 and DeepSeek. Furthermore, the maximum number of repetitions in Step (2) is limited to five. Regardless of whether the last predicted SQL query encounters an execution exception, it is considered the final prediction.

4.3. Evaluation Metrics

The most commonly used evaluation metrics in Text-to-SQL tasks are exact match (EM) and execution accuracy (EX). For EM, the evaluation is based on whether every part of the predicted SQL query matches exactly with the ground truth SQL query. The prediction is considered correct only if all parts of the query are an exact match. For EX, it evaluates the prediction by comparing the execution results of the predicted SQL query and the ground truth SQL query on the same database. A prediction is considered correct only if the execution results are identical.

In spatial databases, the presence of derived columns introduces a challenge where a given question might have multiple valid SQL queries. Additionally, GPT models often exhibit their own distinct style in writing SQL queries, which can result in different SQL queries for the same question. Therefore, when evaluating the methods for spatial databases in Text-to-SQL tasks, the use of the EM metric might be biased, as it requires an exact match of SQL syntax. In contrast, using EX, which directly compares the execution results, is more meaningful and practical. As a result, in our experiments, we exclusively use EX as the evaluation metric.

EX is defined as a percentage, which can be represented as

$$\mathrm{EX} = \frac{|\text{Correctly predicted}|}{|\text{All predicted}|} \times 100\%$$

5. Results

We conduct two sets of experiments based on our developed spatial dataset. In the first set of experiments, we create Dataset 1 by combining the Ada[1], Edu[1], Tourism, and Traffic[1] databases along with their respective samples. Initially, we train on Ada[1] and Edu[1] (100 questions and their corresponding SQL queries) and test on Tourism and Traffic[1] (100 questions and their corresponding ground truth SQL queries), applying all prediction methods. We then reverse the roles, training on Tourism and Traffic[1] (100 questions and their corresponding SQL queries) and testing on Ada[1] and Edu[1] (100 questions and their corresponding ground truth SQL queries). Finally, we evaluate all 200 predictions in Dataset 1 using EX. In this context, training means selecting the top-K questions and their corresponding SQL queries from the training dataset that are most similar to the target question to serve as examples in the prompt.

The second set of experiments follows the same procedure but uses Ada[2], Edu[2], Tourism, and Traffic[2] to form Dataset 2. Specifically, we alternate between training on Ada[2] and Edu[2] (testing on Tourism and Traffic2) and training on Tourism and Traffic[2] (testing on Ada[2] and Edu[2]), again assessing all 200 predictions in Dataset 2.

Although the questions in Dataset 1 and Dataset 2 are identical, the removal of derived columns and related foreign keys in Dataset 2 leads to significant changes in the schemas of the Ada, Edu, and Traffic databases.

5.1. Performance Evaluation of Each Method

For all methods, we conduct experiments with GPT-4 and DeepSeek on both Dataset 1 and Dataset 2, testing scenarios that include no examples in the prompt, as well as scenarios with 1, 3, or 5 examples included. We refer to the scenario without examples in the prompt as 0-shot, while the scenarios with 1, 3, and 5 examples are labeled as 1-shot, 3-shot, and 5-shot, respectively. The results of these experiments are summarized in Tables 3 and 4.

Table 3. Performance (%) of each method with GPT-4.

	Dataset 1				Dataset 2			
Method	**0-Shot**	**1-Shot**	**3-Shot**	**5-Shot**	**0-Shot**	**1-Shot**	**3-Shot**	**5-Shot**
DAIL-SQL	44.5	52.5	59.5	65.5	34.0	47.0	59.0	57.5
SSpa	85.0	83.5	85.0	**87.5**	78.0	80.0	**81.0**	79.0
SSpa-SDBMS	50.0	56.0	61.5	70.5	38.0	48.0	65.5	73.0
SSpa-Geo	80.5	77.5	78.0	77.5	75.0	71.0	71.5	68.5
SSpa-Tips	79.0	80.0	83.5	82.5	74.5	77.5	78.0	75.5

Table 4. Performance (%) of each method with DeepSeek.

	Dataset 1				Dataset 2			
Method	**0-Shot**	**1-Shot**	**3-Shot**	**5-Shot**	**0-Shot**	**1-Shot**	**3-Shot**	**5-Shot**
DAIL-SQL	50.5	63	65.5	70.5	38.0	57.5	62.5	65.5
SSpa	82.5	83	82.5	**86**	75.5	**81**	**81**	**81**
SSpa-SDBMS	51.5	65	72	78.5	41	65.5	76.5	77.5
SSpa-Geo	79.5	79	79	79	72	76	74.5	73.5
SSpa-Tips	80.5	81	80.5	83.5	68.5	77	78.5	76.5

First, we compare the performance differences between the methods SSpa, SSpa-SDBMS, SSpa-Geo, SSpa-Tips, and DAIL-SQL. As illustrated in Tables 3 and 4, the methods SSpa, SSpa-SDBMS, SSpa-Geo, and SSpa-Tips consistently outperform the baseline method DAIL-SQL across both Dataset 1 and Dataset 2, whether using GPT-4 or DeepSeek. Specifically, the SSpa method with GPT-4 achieves the highest performance of 87.5% on Dataset 1

in the 5-shot scenario and 81% on Dataset 2 in the 3-shot scenario, surpassing DAIL-SQL by 22% in both scenarios. Similarly, the SSpa method with DeepSeek attains a peak performance of 86% on Dataset 1 in the 5-shot scenario and 81% on Dataset 2, outperforming DAIL-SQL by at least 15.5% in both scenarios. These performance differences align with our expectations, as DAIL-SQL is specifically designed for relational databases, while spatial databases have their own unique characteristics. The SSpa method and its variants (SSpa-SDBMS, SSpa-Geo, and SSpa-Tips) enhance performance by integrating specific knowledge about the used spatial database management, geographic descriptions, or schema tips into the model's prompt, thereby improving their effectiveness in Text-to-SQL tasks for spatial database.

Next, we examine the performance differences of the SSpa and DAIL-SQL methods across Dataset 1 and Dataset 2. As shown in Tables 3 and 4, both methods (SSpa and DAIL-SQL) consistently perform better on Dataset 1 than on Dataset 2 across all scenarios (0-shot, 1-shot, 3-shot, and 5-shot), whether using GPT-4 or DeepSeek. This result can be attributed to the differences in SQL queries between Dataset 1 and Dataset 2. In Dataset 1, due to the presence of derived columns, the questions can be addressed using both general SQL queries and spatial SQL queries. However, in Dataset 2, where derived columns are removed, the questions can only be answered through spatial SQL queries, which are generally more challenging than general SQL queries. Additionally, compared to relational databases, spatial databases provide less available knowledge, causing GPT models to leverage less spatial database knowledge during training, potentially leading to poorer performance in generating spatial SQL queries than general SQL queries. This highlights that, despite the availability of many GPT-based Text-to-SQL methods for relational databases, significant challenges remain when applying them to Text-to-SQL tasks for spatial databases.

Finally, we evaluate the performance of the SSpa method across various scenarios (0-shot, 1-shot, 3-shot, and 5-shot). As shown in Figure 4, for both GPT-4 and DeepSeek, SSpa achieves optimal performance in non-0-shot scenarios, suggesting that incorporating relevant examples in the prompt can enhance SQL query prediction accuracy. However, increasing the number of examples does not consistently improve performance. For instance, with GPT-4, SSpa outperforms 1-shot with 0-shot on Dataset 1 and 5-shot with 3-shot on Dataset 2. The former may result from a poorly chosen or unrepresentative single example, while the latter could stem from interference among examples in the 5-shot scenario. In contrast, DeepSeek does not exhibit these anomalies, likely due to differences in model architecture and training. Regardless, both GPT-4 and DeepSeek achieve their best results in non-0-shot scenarios.

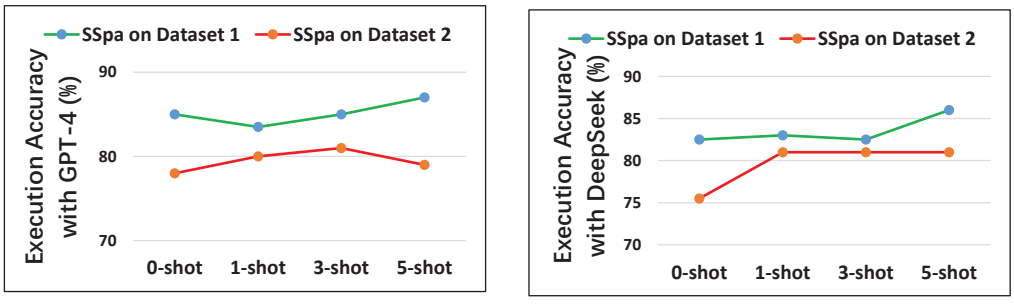


Figure 4. A line chart providing a clear visualization of the performance of SSpa with 0-shot, 1-shot, 3-shot, and 5-shot on both Dataset 1 and Dataset 2. (**a**) Performance of SSpa with GPT-4. (**b**) Performance of SSpa with DeepSeek.

5.2. Effect of Each Component in the Prompt

As shown in Tables 3 and 4, our method, SSpa, outperforms SSpa-Tips, SSpa-SDBMS, and SSpa-Geo across all scenarios (0-shot, 1-shot, 3-shot, and 5-shot), whether using GPT-4 or DeepSeek. This indicates that removing any component from the prompt in our method negatively impacts its performance.

First, we investigate the role of the schema tips component in the prompts for the SSpa, SSpa-SDBMS, and SSpa-Geo methods. In our framework, if an exception occurs during the execution of a GPT-generated SQL query, we extract schema tips from the exception feedback. The extracted schema tips are then incorporated into the original prompt, and the augmented prompt is resubmitted to GPT for a new prediction. Hence, we analyze the number of initially failed queries that were corrected after the inclusion of schema tips. For example, a prompt for the question "What is the total area of the three northeastern provinces?" is sent to GPT, which generates the SQL query "Select Sum (Shape_Area) from cities where province in ('辽宁省', '吉林省', '黑龙江省')". When executed, the SQL query produces an error "no such column: Shape_Area". We extract the schema tip "The column 'Shape_Area' does not exist in the table 'cities' and should not be used.", incorporate it into the original prompt, and resend it to GPT. After iterations, GPT produces the correct SQL query "Select Sum (Area (cities.Shape, 1)) from cities where province in ('辽宁省', '吉林省', '黑龙江省')". Detailed data are presented in Tables 5 and 6.

We observe that, while SSpa-SDBMS with GPT-4 in the 1-shot scenario shows no improvement in correct SQL query predictions on Dataset 1, incorporating the schema tips component increases prediction accuracy in all other scenarios. Notably, with the schema tips component added to the prompt, our SSpa method accurately predicts up to eight SQL queries on both Dataset 1 and Dataset 2, representing 4% of the total 200 predictions. This indicates that in Text-to-SQL tasks for spatial databases, when SQL queries predicted by GPT encounter schema errors, such as missing tables or columns and ambiguous column names, extracting the schema tips and incorporating them into the prompt improves GPT's SQL query prediction accuracy.

Table 5. Number of initially failed SQL query predictions corrected with the adding of the schema tips component by each method using GPT-4.

	Dataset 1				Dataset 2			
Method	**0-Shot**	**1-Shot**	**3-Shot**	**5-Shot**	**0-Shot**	**1-Shot**	**3-Shot**	**5-Shot**
SSpa	1	2	2	8	1	3	7	8
SSpa-Tips	-	-	-	-	-	-	-	-
SSpa-SDBMS	2	0	3	3	1	3	4	4
SSpa-Geo	2	2	2	7	2	3	4	6

Table 6. Number of initially failed SQL query predictions corrected with the adding of the schema tips component by each method using DeepSeek.

	Dataset 1				Dataset 2			
Method	**0-Shot**	**1-Shot**	**3-Shot**	**5-Shot**	**0-Shot**	**1-Shot**	**3-Shot**	**5-Shot**
SSpa	1	2	2	3	1	1	7	8
SSpa-Tips	-	-	-	-	-	-	-	-
SSpa-SDBMS	1	3	7	5	1	5	8	9
SSpa-Geo	2	4	3	5	1	2	7	12

Next, we examine the impact of the geographic descriptions component. The geographic descriptions component refers to the inclusion of geographic region descriptions for each table in the prompts of the SSpa method. As shown in Figure 3, the 'airports' table

is described as "The 'airports' table records information about various airports in China.". In contrast, the prompts for the SSpa-Geo and DAIL-SQL methods do not include such geographic descriptions. As shown in Table 2, 34 out of the 200 questions are related to geographic regions. Therefore, we evaluate the number of correct SQL query predictions for questions related to geographic regions across each method. The detailed results are presented in Tables 7 and 8.

We observe that SSpa consistently outperforms the DAIL-SQL method in the number of correct SQL query predictions across all scenarios. Except in the zero-shot scenario, where SSpa with GPT-4 slightly underperforms SSpa-Geo (SSpa accurately predicts 31 such questions compared to SSpa-Geo's 32), SSpa surpasses SSpa-Geo in all other scenarios. This suggests that for region-specific questions, which frequently arise, incorporating geographic region descriptions into the prompt enhances the GPT model's SQL query prediction accuracy.

Table 7. Number of correct SQL query predictions for region-related questions by each method using GPT-4.

	Dataset 1				Dataset 2			
Method	**0-Shot**	**1-Shot**	**3-Shot**	**5-Shot**	**0-Shot**	**1-Shot**	**3-Shot**	**5-Shot**
DAIL-SQL	9	8	10	11	6	7	11	11
SSpa	32	31	32	31	31	30	32	32
SSpa-Geo	27	16	13	13	32	17	12	11

Table 8. Number of correct SQL query predictions for region-related questions by each method using DeepSeek.

	Dataset 1				Dataset 2			
Method	**0-Shot**	**1-Shot**	**3-Shot**	**5-Shot**	**0-Shot**	**1-Shot**	**3-Shot**	**5-Shot**
DAIL-SQL	10	16	15	16	6	18	17	18
SSpa	31	33	33	33	31	34	33	33
SSpa-SDBMS	13	21	23	24	10	23	31	31
SSpa-Geo	29	23	19	19	28	22	20	18
SSpa-Tips	30	31	31	31	27	32	31	30

Finally, we analyze the role of the SDBMS knowledge component. Compared to the SSpa-SDBMS method, the SSpa method incorporates SDBMS knowledge, specifically descriptions of certain functions, into its prompt. Consequently, we evaluate the number of initially failed SQL query predictions by SSpa-SDBMS that were successfully corrected by incorporating the SDBMS knowledge component (i.e., the SSpa method), with detailed data presented in Table 9.

We observe that across all scenarios (0-shot, 1-shot, 3-shot, and 5-shot), a significant number of initially failed SQL query predictions are successfully corrected after incorporating the SDBMS knowledge component into the prompt. The number of such successful corrections is at least 19 (9.5% of the total 200 queries) for GPT-4 and 14 (7%) for DeepSeek in the 5-shot scenario. This can be attributed to the lack of knowledge about the spatial database used in SSpa-SDBMS. Since spatial database management systems vary in type, each with different functions, the SQL queries generated by SSpa-SDBMS are more likely to invoke functions that are incompatible with the used spatial database management system. For example, we look into the prompts generated by the SSpa-SDBMS and SSpa methods with GPT-4 in the 5-shot scenario for the question "What is the length of the Yangtze River within Hubei Province?". Neither prompt includes examples of the function GLength (c Curve, 1). The SSpa-SDBMS' prompt lacks the SDBMS knowledge, whereas the SSpa' prompt incorpo-

rats the SDBMS knowledge, including an explanation of the GLength (c Curve, 1) function. For this question, the SSpa method produces a correct SQL query "Select Sum (GLength (Intersection (rivers.shape, provinces.shape), 1)) AS LengthInHubei from rivers inner join provinces on Intersects (rivers.shape, provinces.shape) = 1 where rivers.name = '长江' and provinces.name = '湖北省'". In contrast, the SSpa-SDBMS method generates an incorrect SQL query "Select Sum (GLength (Intersection (rivers.shape, provinces.shape))) AS Length from rivers join provinces on Intersects (rivers.shape, provinces.shape) where rivers.name = '长江' and provinces.name = '湖北省'". This query incorrectly uses the GLength (c Curve) function without the required parameter 1. Notably, despite the absence of examples for the GLength (c Curve) or GLength (c Curve, 1) functions in the SSpa-SDBMS prompt, the method still generates a query using GLength (c Curve), albeit incorrectly. This suggests that the inclusion of the GLength (c Curve, 1) explanation in the SSpa prompt effectively can guide GPT to use the correct function format. However, incorporating the SDBMS knowledge into the prompt can occasionally lead to errors in previously correct predictions. Overall, adding the SDBMS knowledge to the prompt significantly enhances the accuracy of GPT's SQL query predictions.

Table 9. Number of initially failed SQL query predictions by SSpa-SDBMS corrected with the SDBMS knowledge component.

	Dataset 1				Dataset 2			
Model	**0-Shot**	**1-Shot**	**3-Shot**	**5-Shot**	**0-Shot**	**1-Shot**	**3-Shot**	**5-Shot**
GPT-4	73	57	49	36	86	66	35	19
DeepSeek	66	38	32	21	78	35	17	14

6. Limitations

There are also several limitations in our work.

First, in the SDBMS knowledge component of the prompt, we manually specify only a few commonly used functions. The accuracy of the generated SQL queries could likely be improved if more relevant functions, tailored to each specific question, were provided. Therefore, in future work, we will automatically select and include appropriate functions in the prompt based on the target question.

Second, in the schema tips component of the prompt, we address only three scenarios: non-existent tables or columns and column name ambiguities. By extracting a broader range of schema tips from the exception feedback and incorporating them into the prompt, we believe the accuracy of the generated SQL queries could be further enhanced.

Finally, due to the inherent difficulty in constructing spatial questions and their corresponding SQL queries, our dataset remains limited. As a result, we have identified only a few patterns, such as adding geographic region descriptions for tables. With a larger dataset and a more diverse set of question types, we believe additional patterns more suitable for constructing prompts for spatial databases could be discovered. Of course, all three of these limitations represent areas for future work and can be gradually addressed over time.

7. Conclusions

Text-to-SQL for spatial databases enables the translation of natural language questions into corresponding SQL queries based on a given database schema, making spatial data more accessible to non-experts. This capability has garnered increasing attention from researchers. In our study, we propose a GPT-based method for constructing prompts to solve the Text-to-SQL task for spatial databases. Our method introduces three key innovations: the integration of SDBMS knowledge, geographic descriptions, and schema

tips within the prompt. We detail the methodology and present systematic experiments. The results demonstrate that all three components, including SDBMS knowledge, geographic descriptions, and schema tips, are essential for optimal performance. Omitting any of these components adversely affects the accuracy of the method. Additionally, to further advance Text-to-SQL for spatial databases, we have created a bilingual dataset that supports both Chinese and English. Our work has the potential to greatly enhance the accessibility of spatial databases for non-experts, facilitating their use and application.

Author Contributions: Conceptualization, Hui Wang and Li Guo; methodology, Hui Wang and Li Guo; software, Le Liu and Hui Wang; data curation, Le Liu; writing—original draft preparation, Hui Wang and Li Guo; writing—review and editing, Hui Wang, Li Guo, Yubin Liang, and Jiajin Huang; supervision, Hui Wang and Li Guo; funding acquisition, Hui Wang. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Doctoral Fund of Tianjin Normal University (Grant No. 52XB1619) and the Open Fund of Tianjin Geospatial Information Technology Engineering Center.

Data Availability Statement: The data that support the findings of this study are openly available in GitHub at https://github.com/beta512/SpatialSQL (accessed on 14 July 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Leszczynski, A.; Crampton, J. Introduction: Spatial Big Data and Everyday Life. *Big Data Soc.* **2016**, *3*, 1–6. [CrossRef]
2. Liu, M.; Wang, X.; Xu, J. NALSD: A Natural Language Interface for Spatial Databases. In Proceedings of the 18th International Symposium on Spatial and Temporal Data, Calgary, AB, Canada, 23–25 August 2023; pp. 175–179.
3. Liu, M.; Wang, X.; Xu, J.; Lu, H. NALSpatial: An Effective Natural Language Transformation Framework for Queries over Spatial Data. In Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems, Hamburg, Germany, 13–16 November 2023; p. 4.
4. Wang, X.; Liu, M.; Xu, J.; Lu, H. NALMO: Transforming Queries in Natural Language for Moving Objects Databases. *GeoInformatica* **2023**, *27*, 427–460. [CrossRef]
5. Liu, J.; Shen, D.; Zhang, Y.; Dolan, B.; Carin, L.; Chen, W. What Makes Good In-Context Examples for GPT-3? In Proceedings of the 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures, Dublin, Ireland, 26–27 May 2022; pp. 100–114.
6. Guo, C.; Tian, Z.; Tang, J.; Wang, P.; Wen, Z.; Yang, K.; Wang, T. Prompting GPT-3.5 for Text-to-SQL with De-semanticization and Skeleton Retrieval. In Proceedings of the Trends in Artificial Intelligence—20th Pacific Rim International Conference on Artificial Intelligence, Jakarta, Indonesia, 15–19 November 2023; pp. 262–274.
7. Pourreza, M.; Rafiei, D. 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, 10–16 December 2024; pp. 36339–36348.
8. Zelle, J.M.; Mooney, R.J. Learning to Parse Database Queries using Inductive Logic Programming. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, OR, USA, 4–8 August 1996; pp. 1050–1055.
9. Tang, L.R.; Mooney, R.J. Automated Construction of Database Interfaces: Intergrating Statistical and Relational Learning for Semantic Parsing. In Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, Hong Kong, China, 7–8 October 2000; pp. 133–141.
10. Iyer, S.; Konstas, I.; Cheung, A.; Krishnamurthy, J.; Zettlemoyer, L. Learning a Neural Semantic Parser from User Feedback. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, BC, Canada, 30 July–4 August 2017; pp. 963–973.
11. Zhong, V.; Xiong, C.; Socher, R. Seq2SQL: Generating Structured Queries from Natural Language Using Reinforcement Learning. *arXiv* **2017**, arXiv:1709.00103. [CrossRef]
12. Yu, T.; Zhang, R.; Yang, K.; Yasunaga, M.; Wang, D.; Li, Z.; Ma, J.; Li, I.; Yao, Q.; Roman, S.; et al. 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, Brussels, Belgium, 31 October–4 November 2018; pp. 3911–3921.

13. Min, Q.; Shi, Y.; Zhang, Y. A Pilot Study for Chinese SQL Semantic Parsing. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Hong Kong, China, 3–7 November 2019; pp. 3652–3658.
14. Wang, L.; Zhang, A.; Wu, K.; Sun, K.; Li, Z.; Wu, H.; Zhang, M.; Wang, H. DuSQL: A Large-Scale and Pragmatic Chinese Text-to-SQL Dataset. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, Online, 16–20 November 2020; pp. 6923–6935.
15. Gan, Y.; Chen, X.; Huang, Q.; Purver, M.; Woodward, J.R.; Xie, J.; Huang, P. 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, Online, 1–6 August 2021; pp. 2505–2515.
16. Gan, Y.; Chen, X.; Purver, M. Exploring Underexplored Limitations of Cross-Domain Text-to-SQL Generalization. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, 7–11 November 2021; pp. 8926–8931.
17. Shaw, P.; Chang, M.-W.; Pasupat, P.; Toutanova, K. Compositional Generalization and Natural Language Variation: Can a Semantic Parsing Approach Handle Both? In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Online, 1–6 August 2021; pp. 922–938.
18. Li, J.; Hui, B.; Qu, G.; Yang, J.; Li, B.; Li, B.; Wang, B.; Qin, B.; Geng, R.; Huo, N.; et al. 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, New Orleans, LA, USA, 10–16 December 2023; pp. 42330–42357.
19. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 3104–3112.
20. Guo, J.; Zhan, Z.; Gao, Y.; Xiao, Y.; Lou, J.-G.; Liu, T.; Zhang, D. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 4524–4535.
21. Choi, D.; Shin, M.C.; Kim, E.; Shin, D.R. RYANSQL: Recursively Applying Sketch-Based Slot Fillings for Complex Text-to-SQL in Cross-Domain Databases. *Comput. Linguist.* **2021**, *47*, 309–332. [CrossRef]
22. Hwang, W.; Yim, J.; Park, S.; Seo, M. A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019.
23. Yin, P.; Neubig, G.; Yih, W.-t.; Riedel, S. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 8413–8426.
24. Herzig, J.; Nowak, P.K.; Müller, T.; Piccinno, F.; Eisenschlos, J. TAPAS: Weakly Supervised Table Parsing via Pre-Training. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 4320–4333.
25. Yu, T.; Wu, C.-S.; Lin, X.V.; Wang, B.; Tan, Y.C.; Yang, X.; Radev, D.R.; Socher, R.; Xiong, C. Grappa: Grammar-Augmented Pre-Training for Table Semantic Parsing. In Proceedings of the 9th International Conference on Learning Representations, Online, 3–7 May 2021.
26. Li, J.; Wang, W.; Ku, W.-S.; Tian, Y.; Wang, H. SpatialNLI: A spatial Domain Natural Language Interface to Databases Using Spatial Comprehension. In Proceedings of the 27th ACM International Conference on Advances in Geographic Information Systems, Chicago, IL, USA, 5–8 November 2019; p. 10.
27. Wang, W.; Li, J.; Ku, W.-S.; Wang, H. Multilingual Spatial Domain Natural Language Interface to Databases. *GeoInformatica* **2024**, *28*, 29–52. [CrossRef]
28. Hong, Z.; Yuan, Z.; Zhang, Q.; Chen, H.; Dong, J.; Huang, F.; Huang, X. Next-Generation Database Interfaces: A Survey of LLM-Based Text-to-SQL. *arXiv* **2024**, arXiv:2406.08426.
29. Liu, X.; Shen, S.; Li, B.; Ma, P.; Jiang, R.; Zhang, Y.; Fan, J.; Li, G.; Tang, N.; Luo, Y. A Survey of NL2SQL with Large Language Models: Where Are We, and Where Are We Going? *arXiv* **2024**, arXiv:2408.05109. [CrossRef]
30. Shi, L.; Tang, Z.; Zhang, N.; Zhang, X.; Yang, Z. A Survey on Employing Large Language Models for Text-to-SQL Tasks. *arXiv* **2024**, arXiv:2407.15186. [CrossRef]
31. Dong, X.; Zhang, C.; Ge, Y.; Mao, Y.; Gao, Y.; Chen, L.; Lin, J.; Lou, D. C3: Zero-Shot Text-to-SQL with ChatGPT. *arXiv* **2023**, arXiv:2307.07306.
32. Zhang, H.; Cao, R.; Chen, L.; Xu, H.; Yu, K. ACT-SQL: In-Context Learning for Text-to-SQL with Automatically-Generated Chain-of-Thought. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, 6–10 December 2023; pp. 3501–3532.
33. Gao, D.; Wang, H.; Li, Y.; Sun, X.; Qian, Y.; Ding, B.; Zhou, J. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *Proc. Vldb Endow.* **2024**, *17*, 1132–1145. [CrossRef]

34. Wu, H.; Shen, Z.; Hou, S.; Liang, J.; Jiao, H.; Qing, Y.; Zhang, X.; Li, X.; Gui, Z.; Guan, X.; et al. AutoGEEval: A Multimodal and Automated Evaluation Framework for Geospatial Code Generation on GEE with Large Language Models. *ISPRS Int. J.-Geo-Inf.* **2025**, *14*, 256. [CrossRef]
35. Hou, S.; Jiao, H.; Shen, Z.; Liang, J.; Zhao, A.; Zhang, X.; Wang, J.; Wu, H. Chain-of-Programming (CoP): Empowering Large Language Models for Geospatial Code Generation Task. *Int. J. Digit. Earth* **2025**, *18*, 2509812. [CrossRef]
36. Hou, S.; Liang, J.; Zhao, A.; Wu, H. GEE-OPs: An Operator Knowledge Base for Geospatial Code Generation on the Google Earth Engine Platform Powered by Large Language Models. *Geo-Spat. Inf. Sci.* **2025**, 1–22. [CrossRef]
37. Hou, S.; Shen, Z.; Zhao, A.; Liang, J.; Gui, Z.; Guan, X.; Li, R.; Wu, H. GeoCode-GPT: A Large Language Model for Geospatial Code Generation. *Int. J. Appl. Earth Obs. Geoinf.* **2025**, *138*, 104456. [CrossRef]
38. Hou, S.; Zhao, A.; Liang, J.; Shen, Z.; Wu, H. Geo-FuB: A Method for Constructing An Operator-Function Knowledge Base for Geospatial Code Generation with Large Language Models. *Knowl.-Based Syst.* **2025**, *319*, 113624. [CrossRef]
39. Jiang, Y.; Yang, C. Is ChatGPT a Good Geospatial Data Analyst? Exploring the Integration of Natural Language into Structured Query Language within a Spatial Database. *ISPRS Int. J.-Geo-Inf.* **2024**, *13*, 26. [CrossRef]
40. Nan, L.; Zhao, Y.; Zou, W.; Ri, N.; Tae, J.; Zhang, E.; Cohan, A.; Radev, D. Enhancing Text-to-SQL Capabilities of Large Language Models: A Study on Prompt Design Strategies. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, 6–10 December 2023; pp. 14935–14956.
41. Chang, S.; Fosler-Lussier, E. How to Prompt LLMs for Text-to-SQL: A Study in Zero-Shot, Single-Domain, and Cross-Domain Settings. *arXiv* **2023**, arXiv:2305.11853.