INSTANT: Inquiring Schema Linking and Encoding to boost Text to SQL Parsing from Pre-trained Models

A

report submitted in fulfillment for the award of the degree of

Integrated Post Graduate

in

Information Technology

Masters of Technology Project

Final Evaluation Report

By

Akshay Khatri: 2018IMT-013

Under the Supervision of

Dr. SUNIL KUMAR

Department of Information Technology

ABV-INDIAN INSTITUTE OF INFORMATION TECHNOLOGY
AND MANAGEMENT GWALIOR

GWALIOR, INDIA

DECLARATION

I hereby certify that the work, which is being presented in the report/thesis, entitled **INSTANT: Inquiring Schema Linking and Encoding to boost Text to SQL Parsing from Pre-trained Models** , in fulfillment of the requirement for the award of the degree of **Integrated Post Graduate - Master of Technology in Information Technology** and submitted to the institution is an authentic record of my/our own work carried out during the period July-2022 to May-2023 under the supervision of **Dr. Sunil Kumar**. I also cited the reference about the text(s)/figure(s)/table(s) from where they have been taken.

Dated: **Signature of the candidate**

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dated: **Signature of supervisor**

Acknowledgements

I would like to express my sincere thanks to everyone who made this seminar report possible. First and foremost, I would like to express my profound gratitude and respect to my supervisor, Dr. Sunil Kumar, who has been guiding force behind this work. I am immensely grateful for his invaluable guidance, constant encouragement, and insightful feedback on my work. I am fortunate to have had an advisor who provided me with the freedom to think independently and explore new ideas. His patience and attention to detail in reviewing and commenting on my manuscripts, as well as countless revisions of this dissertation, have been a great help to me. His commitment and dedication to research have been and will continue to be a constant source of inspiration. I have no doubt that completing my degree in a proper and timely manner would have been impossible without his support. I am truly privileged to have had the opportunity to work with such a wonderful person.

Finally, I would like to express my gratitude to the Almighty God for providing me with this opportunity and for blessing me with the strength to overcome all obstacles and achieve success.

Akshay Khatri

Abstract

The need of developing text-to-SQL parsers that can be applied to latest databases has been recognised, and schema linking, i.e. correctly recognising mentions of unseen columns or tables when generating SQLs, is a vital step towards achieving this aim. Recent studies suggest that current domain generalization techniques are limited. Additionally, the methods utilized by existing neural semantic parsers to handle schema linking are not effective in more practical scenarios.

In this study, we introduce a novel framework for extracting relational structures from large pre-trained language models (PLMs) using a probing technique based on the Poincaré distance metric. The induced relations are then utilized to enhance current graph-based parsers, leading to improved schema linking. Our probing approach is demonstrated to be more effective than commonly-used rule-based methods for capturing semantic correspondences, particularly in cases where entities and surface forms of mentions differ. Moreover, our probing technique is entirely unsupervised and does not require additional parameters. Extensive experiments indicate that our framework achieves great performance on three benchmark datasets. Through qualitative analysis, we verify that our probing technique can effectively identify the desired relational structures. The results demonstrate that utilizing semantic and relational knowledge from pre-trained language models can improve the ability of Text to SQL parsers to comprehend domain knowledge and generalize to new domains.

Contents

List of Figures

List of Tables

1

Introduction

This section presents introduction in the area of natural language processing, and relational database queries from a given natural language that is pertinent. Followed by the motivation to do thesis work and with the problem statement.

1.1 Introduction

The aim of text to SQL parsing is to transform natural language (NL) inquiry to structured query language (SQL) equivalent with respect to the relational database. Although experienced professionals may easily access relational databases using handmade SQLs, a natural language interface based on text to SQL parsing could make relational database available to a wide variety of non technical users. Thus the growing interest in text to SQL parsing has attracted the attention of both academic and industrial sectors.

One difficult aim of text to SQL parsing is to achieve domain generalization, i.e., designing parsers which can be easily applied to other databases (or domains). The accessibility of benchmarks has resulted in several breakthroughs in constructing domain-generalized parsers. Schema linkage - appropriately matching entity reference in the natural language query to the schema columns or tables, on unknown databases - is critical to achieve domain generalization. As seen in Figure 1.1, a parser must connect mentions of authors and writers to the relevant column author.

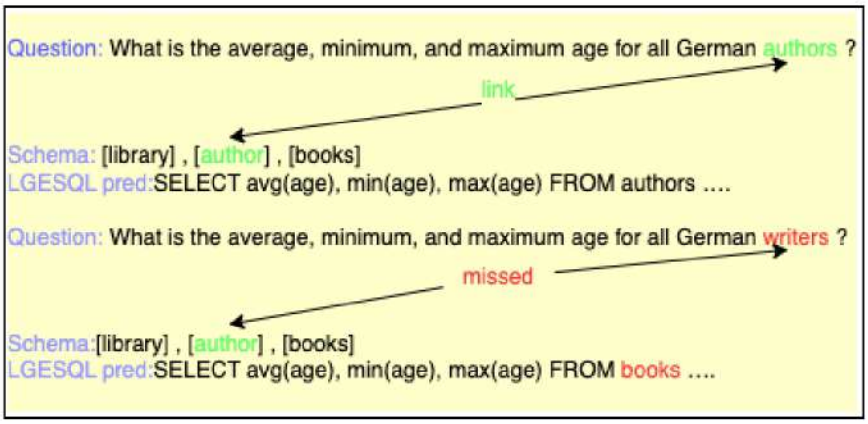

Figure 1.1: The provided diagram depicts the relational structures of schema linking that exist between natural language (NL) queries and database schemas. Exact string matching and other heuristic approaches would fail to capture resemblance when mentioned surface forms differ.

1.2 Natural Language Processing

In the field of computing, natural language (NL) refers to the languages that humans use to communicate with each other [1]. Examples of NL include English, Hindi, Chinese,

Japanese, and many others. These languages are typically different from the languages used to provide commands to computers, which are usually artificially constructed or composed of syntax-specific programming languages. The term "natural language" typically refers to written language, but it may also refer to spoken language. Natural language is often used by humans to express emotions, provide information, and take action.

Every human expression, whether spoken or written, carries a wealth of information. The chosen topics, tone, and words all contribute to the overall message and can be analyzed for valuable insights. However, the challenge arises when dealing with large groups of people or texts. A single person can produce hundreds or even thousands of words in a single speech, each sentence adding its own layer of complexity. Scaling up to analyze hundreds, thousands, or even millions of people or texts in a given location becomes unmanageable.

The advancements in machine learning have enabled the analysis of unstructured data like conversations, speeches, and tweets, which do not fit neatly into the conventional structure of relational databases. Instead of relying on keyword-based approaches, which only scratch the surface, machine learning techniques allow for a deeper understanding of the meaning behind the words. This allows for the identification of figures of speech like sarcasm or irony and the analysis of sentiment, making it easier to manage and extract valuable insights from complex and diverse data.

Natural language processing (NLP) is an interdisciplinary field of study that combines computer science, artificial intelligence, data science, and linguistics to enable computers to read, understand, and generate human languages. It involves the development of algorithms and models that can analyze and interpret large amounts of natural language data, allowing for a range of applications such as language translation, sentiment analysis, text summarization, and more. NLP has been flourishing due to the widespread availability of databases and the significant improvements in computational power, granting practitioners the ability to accomplish remarkable results in various areas such as health care, communications, economics, human resources, and more.

In simpler terms, NLP involves the automated processing of human language in spoken or written form. While this concept alone is fascinating, the true value of the technology lies in its practical applications. NLP techniques can be applied to various aspects of our lives, and the range of its applications is expanding rapidly. Here are some examples:

- NLP plays a crucial role in healthcare by enabling the identification and prediction of various health conditions using computerized fitness reports. This technology is being investigated in a range of health situations such as heart-related diseases, neurological disorders, and mental health conditions. Amazon's Comprehend Medical is an example of an application that uses NLP to take disease conditions and symptoms, medications, and treatment results from various electronic health records including patient records and clinical examination reports. This technology is helping healthcare professionals to make better-informed decisions and improve patient outcomes.

- To elaborate, NLP can be used to analyze customer reviews, social media posts, and other forms of electronic communication to identify patterns in consumer feedback. This information can be used by institutions to improve their services or products, adjust their marketing strategies, and ultimately increase customer satisfaction. For example, a company can use NLP to identify common complaints or concerns mentioned by customers and take steps to address those issues. Additionally, NLP can help companies understand what aspects of their products or services are most appealing to customers and use that information to make targeted improvements or promotions.

- IBM's cognitive assistant is an example of a personalized search application that uses NLP to study a person's behavior and preferences to make recommendations on things like books, music, and other items that the person may have difficulty remembering or retrieving from memory. This technology is a powerful tool for individuals who need assistance in finding the information they need quickly and

efficiently.

- NLP techniques can be used to automatically analyze and categorize the content of emails, which can help organizations like Google and Yahoo to efficiently sort and filter incoming messages. These techniques can also be used to detect and flag spam or malicious content, preventing it from reaching the recipient's inbox.

- The NLP Group at MIT has generated a novel system that help in recognizing fake news by determining the validity and political leanings of a news source. The system uses NLP techniques to determine whether a news source can be believed or not.

- Having insights into events and public opinions can be very valuable to traders in the finance industry. NLP techniques are being used to monitor news, articles, and comments about potential company alliances, and all of these insights and features can later be integrated into a trading algorithm to generate significant profits.

- NLP techniques are increasingly being used in various stages of the recruitment process, from analyzing job postings and resumes to conducting interviews and evaluating candidates. For example, NLP can help analyze job postings to identify the required skills and experience, and then compare that with the candidates' resumes to determine the best matches. It can also be used to analyze candidates' responses in interviews to identify specific traits and qualities, such as communication skills or problem-solving abilities.

- LegalMation has developed a program that utilizes NLP technology powered by IBM Watson to automate conventional prosecution tasks, helping legal teams to save time and lower costs while also redirecting their focus towards more important aspects.

1.3 Relational Database

A relational database is a type of database that stores and provide access to stored data objects that are related to each another. Relational databases use a logical model to organize data into tables, providing a straightforward representation of information. Each row in a table represents a record and is assigned a unique identifier, known as the key. The columns of the table store the different attributes of the data, and each record has a value for each attribute, simplifying the establishment of relationships between data points [2].

The relational model separates the structure of the data from its physical storage. This means that changes to the physical storage, such as renaming a database file, do not affect the logical structure of the data, including the tables, records, and indexes. The separation of these two aspects of data management allows for greater flexibility and easier maintenance of the database.

In the world of databases, there is a clear distinction between physical and logical operations. Logical operations refer to the defined actions that allow users to manage and exploit data and data structures within a databases. They enable users to specify the content it requires. In contrast, physical operations determine how the data must be stored, accessed, and manipulated. They focus on the mechanics of the data storage and retrieval process. Together, logical and physical operations ensure that the database functions efficiently and effectively.

Relational databases enforce data integrity rules to certify that data is always accurate and consistent. A simple example of it is, a rule could be set to prevent duplicate rows from being entered into a table, which helps to eliminate the possibility of incorrect data being entered into the database. Other integrity rules may include constraints on data types, ensuring that only valid data is entered into specific fields, and defining relationships between tables to maintain consistency in data across the entire database.

1.3.1 The Relational Model

In the starting stages of databases, each user stored data in a unique relation structure defined by the user itself. This made it challenging for developers to retrieve the data they needed, as they had to understand the structure of each data set. These data-structures were also ineffective and challenging to manage. The relational database model was intended to address these issues by providing a standardized way to store and retrieve data as shown in Figure 1.1. As an example, consider the relationship between teachers working in various departments.

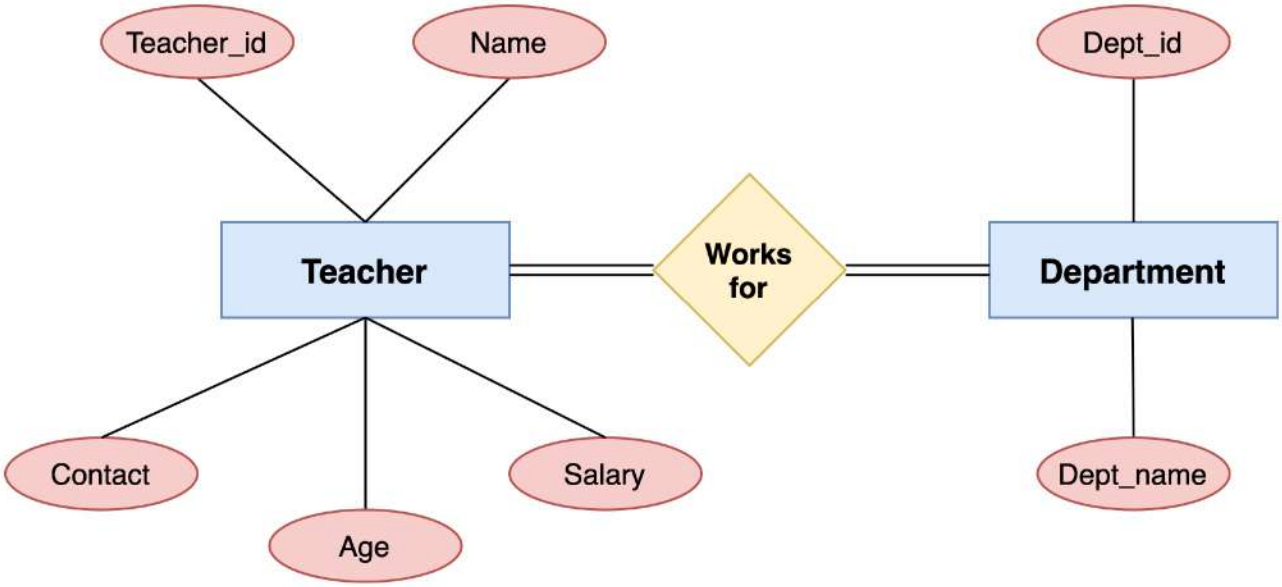

Figure 1.2: A simple 'Relational Model' diagram representing the relation between Teachers and Departments.

The use of tables in the relational database model provides a structured and organized way of storing and retrieving data, making it easy for developers to understand and work with. The tables are made up of rows and columns, with each row representing a unique record and each column representing a specific attribute of the data. This structure allows for efficient querying and manipulation of data, making it a popular choice for storing and managing large amounts of information in various applications.

To add more clarity, Structured Query Language(SQL) is a programming language that is widely used for managing relational databases. It is a declarative language, mean-

ing it allows developers to specify what they want the database to do, rather than how to do it. SQL is based on the relational algebra, which is a mathematical system for manipulating tables and sets of data. Because of its inherent structure and mathematical foundation, SQL queries can be optimized for performance, making it a powerful tool for managing and querying data in relational databases.

1.3.2 Example of Relational Database

Here is a simple illustration of two tables that a small business may use to process product orders. The first table shown in Figure 1.2 contains information related to the customer, such as their name, address, contact information, and billing details. Each piece of information (attribute) is stored in a separate column, and each row is assigned a unique identifying ID (a key). The second table is a customer order table that stores various details related to the order placed by a customer, including the product ordered, quantity, size, color, and other relevant information. However, this table does not contain the name or contact information of the customer; it only includes the ID of the customer who placed the order.

Despite having only one column in common (the key), the relational database can establish a relationship between these two tables. The efficient retrieval and linking of data across different tables in a relational database is one of the key advantages of this type of database. In the example given, the relational database enables the company's order processing application to quickly access the necessary information from multiple tables to fulfil an order. This streamlined process helps to ensure that the customer receives their order on time and the company receives payment in a timely manner, improving overall efficiency and customer satisfaction.

1.3.3 Advantages of a Relational Database

Organizations of all types and sizes continue to rely on the simple yet powerful relational model for a wide range of data needs. Relational databases are used for tracking and

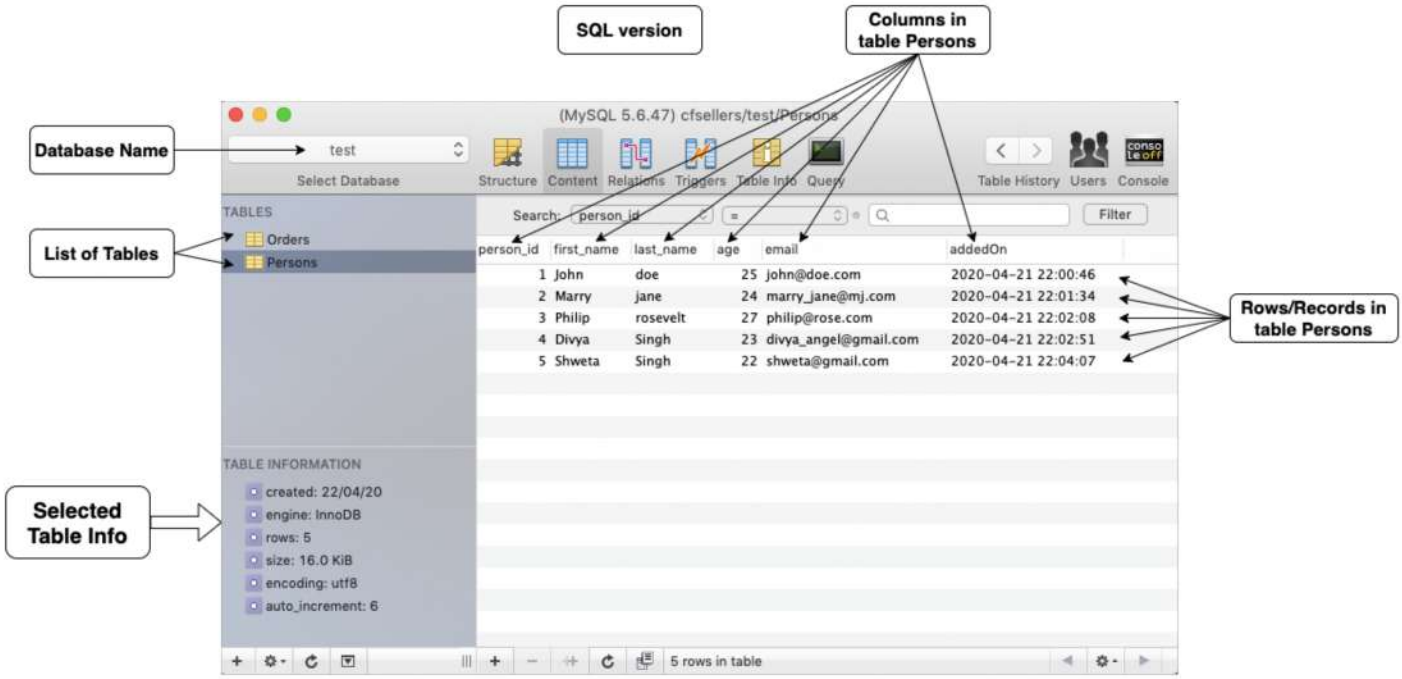

Figure 1.3: A simple illustration of relational database in Sequel Pro software depicting the columns and tables in the database.

managing inventories, processing e-commerce transactions, maintaining large amounts of mission-critical customer information, and much more. Any data requiring objects that relate to one another and need to be managed in a secure, rule-based, and consistent manner can be considered for a relational database.

1.3.4 ACID properties

The four fundamental characteristics that describe a relational database's transactional event are atomicity, isolation, consistency, and durability, commonly abbreviated as ACID.

- **Atomicity** refers to the concept that all components of a database transaction are treated as a single, indivisible unit.
- **Consistency** describes the rules for ensuring that data objects are in a correct state after a transaction event.
- **Isolation** ensures that the outcome of a transaction event is hidden from others

until it is ultimately committed, in order to prevent any confusion.

- **Durability** ensures that the changes made to the data become permanent once a transaction has been committed, and are not lost due to any subsequent failures.

1.4 Structured Querying Language (SQL)

Semantic Querying Language, or SQL, is a domain-specific programming language used to manage and manipulate relational databases. SQL is used to create and modify database schemas, insert, update, and delete records in a database, and retrieve data from a database [3]. Relational databases are widely used to maintain data consistency, and they offer significant benefits such as power, performance, flexibility, and the ability to support new hardware technologies.

Information is at the heart of many web and mobile applications. For example, a social media application like Facebook stores information about user profiles, their stories, and posts. To handle this large amount of data, a database system is needed. SQL helps developers to interact with this stored data in database system. Figure 1.4 shows the result of a simple 'SELECT' query.

SQL, or Structured Query Language, is a powerful tool used to manage and query relational databases. These databases are popular due to their ability to maintain data consistency, flexibility, and support for new hardware technologies, as well as their performance. SQL was first standardized by the American National Standards Institute (ANSI) in 1986, and then by the International Organization for Standardization (ISO) in 1987, cementing its place as a universal language for working with relational databases.

Here are some key features of SQL:

- SQL is used to interact with databases, and can be used to retrieve, insert, update, and delete data in tables.

- SQL allows for the creation and modification of database schemas and structures.

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

```
SELECT CustomerName,City FROM Customers;
```

CustomerName	City
Alfreds Futterkiste	Berlin
Ana Trujillo Emparedados y helados	México D.F.
Antonio Moreno Taquería	México D.F.
Around the Horn	London
Berglunds snabbköp	Luleå
Blauer See Delikatessen	Mannheim

Figure 1.4: Evample of SQL query of type 'SELECT column FROM table'.

- SQL supports various types of database constraints, such as primary keys, foreign keys, and unique constraints, to maintain data integrity.
- SQL provides powerful tools for data aggregation, filtering, and sorting, allowing for complex queries to be constructed.
- SQL is widely used and supported, with many different implementations and vendors providing their own variations and extensions.
- SQL is studied as a 4th-generation language (4GL), where as C++ and Java are 3rd-generation languages (3GLs).

Semantic parsing is the process of converting natural language text into a representation of formal meaning, such as logical forms or structured queries. Recently, there has been a surge in interest in developing machine learning techniques for semantic parsing, aided in part by the availability of large-scale corpora consisting of utterances annotated with formal meaning representations.

However, many prior systems in text to SQL parsing rely on predefined template and manually constructed features to predict the correct logical form for a given statement. This approach often results in parsing models that are limited to specific domains or representations.

1.5 Motivation

Relational databases are popularly used for their ability to maintain consistency, provide high performance, flexibility, and power. Querying databases through natural language can greatly benefit users from diverse backgrounds in analyzing large amounts of data. However, most end users are not familiar with the syntax and intricacies of querying languages for databases, and existing systems that address this issue are not very robust. By enabling efficient querying of relational databases through natural language instructions, users can obtain useful information without having to learn complex querying languages. This project aims to achieve this task.

1.6 Problem Statement

Converting natural language text to SQL requires capturing and processing a lot of details to ensure accurate translation. However, there are currently issues in dealing with text that is not in the correct format and in handling out-of-domain words. Additionally, the large gap in syntax between natural language and SQL presents challenges in accurately mapping between the two. Therefore, there is a need to propose a more efficient solution to address these problems.

2

Literature Review

This section presents recent or ongoing work in the area of generating relational database query from a given natural language that are pertinent. The methodologies employed in diverse research are evaluated critically to determine the advantages and disadvantages of using a specific approach.

2.1 Background

There has been significant research into translating natural language into SQL queries, but early efforts were often specific to particular databases and required significant customization for each new database. This research is important for giving people access to the vast amount of data available in databases and enabling them to retrieve information without having to understand the complexities of Syntactic Query Language. However, constructing a complex query using SQL requires expertise, so the aim of research on facilitating natural language to query construction is to minimize the complexity of the query constructed.

2.2 Key Related Research

Recent research works have attempted to address the issue mentioned in section 2.1 by incorporating feedback from users [4–7]. However, this approach is dependent on human input and can be time-consuming. Another approach involves using methods proposed by [8, 9] as an additional input. However, this approach may face scalability and privacy concerns when dealing with large-scale user databases.

SQLizer is a method that aims to address the same problem discussed earlier, and it is not limited to any particular database [10]. It is a sketch-based approach that relies on an off-the-shelf semantic parser to convert natural language queries into sketches [10], which are then refined using techniques involving programming languages such as type-directed sketch completion and automatic repair [11, 12]. However, because SQLizer does not involve any database-specific training and its software is not publicly available, its performance on the WikiSQL dataset is uncertain. Therefore, recent research on natural language to SQL conversion has shifted towards neural network-based methods [13].

Seq2SQL is considered the most appropriate method for natural language to SQL translation and has achieved state-of-the-art performance on the WikiSQL dataset [14]. The suggested model uses a strategy to generate the query which includes unordered

components that are less suitable for cross-entropy loss optimization. Seq2SQL also uses SQL's framework to prune the space of generated queries and significantly simplify the generation issue [15, 16]. Using policy-based reinforcement learning techniques to execute queries in the WikiSQL environment, Seq2SQL outperforms a state-of-the-art semantic parser, improving accuracy from 35.9% to 59.4% and logical form precision from 23.4% to 48.3% [17].

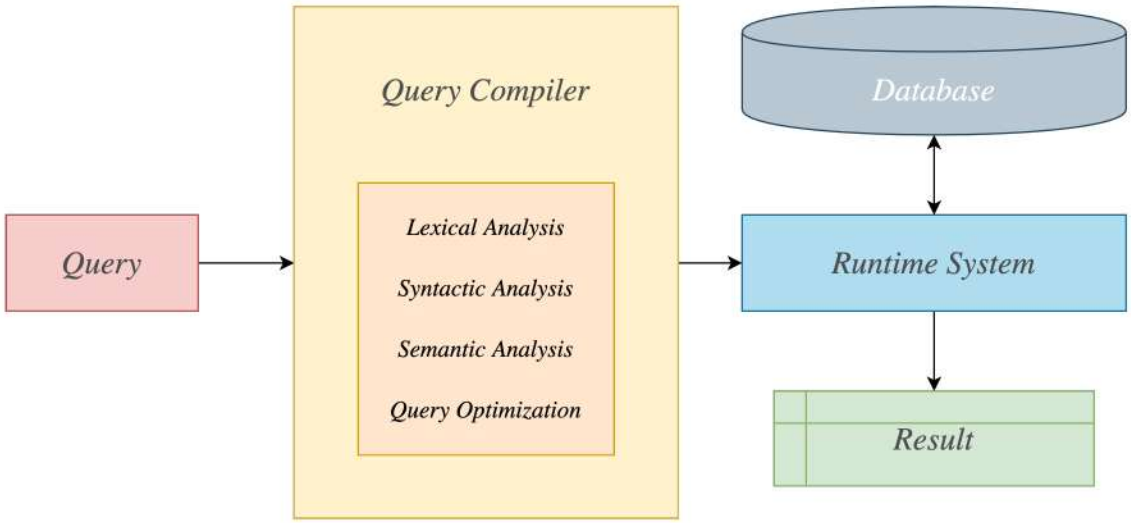

Figure 2.1: A diagram depicts the processing of queries in relational databases. The query is first analyzed by the "Query Compiler", which then retrieves the results from the database during runtime.

Referring to the broader problem of semantic parsing, the challenge of converting natural language into SQL queries can be viewed as a specific example [18]. Numerous studies have focused on the logical parsing of natural language descriptions, but they do not address the task of SQL generation. Furthermore, most of these studies necessitate customization for the specific domain and are not generally applicable.

Dong and M. Lapata [19] proposed a sequence-to-tree based model as a generic approach for semantic parsing, which has shown to produce state-of-the-art results in many tasks. However, in [14], it was found to be less efficient compared to the Seq2SQL approach. Additionally, [20] focused on hierarchical semantic parsing.

One of the main challenges when using neural network models for natural language to SQL query generation is the availability of large datasets containing question-query pairs.

Recent works, such as [14, 21], have addressed this issue by generating datasets using templates to randomly produce queries and corresponding natural language questions, which are then refined by human evaluators to ensure proper English syntax. Another approach is to use feedback-based learning through reinforcement learning (RL), where the system alternates between making predictions and training based on user feedback on correctness [4]. Other approaches, such as end-to-end procedures, have also been explored to avoid the data bottleneck [18, 22]. One significant contribution of this thesis is the standardization of various datasets to address the issue of limited data resources.

2.3 Analysis

In recent years, the Text to SQL field, which involves generating SQL queries from natural language questions, has received significant attention. End-to-end approaches have achieved over 80% accuracy on various public Text to SQL benchmarks, such as ATIS, GeoQuery, and WikiSQL. However, the top-performing solutions have been found to perform unsatisfactorily on the newly acquired cross-domain Text to SQL benchmark dataset, Spider [23].

Text to SQL Parsing.

(i) Lately, significant advances occurred on the encoder [24–26], decoder [27, 28] and table-based pre-training models [29–31] on Spider benchmark [23].

(ii) To address this issue, Yu et al. [31] introduced additional input features that incorporate prior knowledge of column types and schema linking.

(iii) Guo et al. [32] employed heuristic rules to construct an intermediate representation for Text to SQL, and then applied a pointer-generator network to produce the SQL query. Meanwhile, Rui et al. [33] proposed the EditSQL model, which uses a coattention mechanism to calculate the corelation between natural language token and

schema token, and then utilizes a sequence-to-sequence model with copy mechanism to generate the SQL query.

(iv) The RAT-SQL method, proposed by Bailin et al. [34], is a recent approach that uses relational-graph attention to control various predefined relations. It also considers both edge features in order to improve the accuracy of the generated SQL queries from natural language questions.

(v) Scholak et al. propose a novel approach called PICARD [35] that aims to improve the decoding process of language models through incremental parsing. The method is based on the idea of constraining auto-regressive decoders with intermediate syntactic and semantic structures of the output sequence.

2.4 Research Gap

Observed from the literature review and analysis, we can see the following research gap can be seen and addressed. There comes challenges that occur between intents conveyed in natural language and specifics of execution in SQL. This difficulty is seen as a question of inconsistency.

(i) The challenge in Text to SQL lies in the fact that SQL is designed for efficient querying of relational databases, rather than reflecting the meaning of natural language. As a result, there is a natural mismatch between the intentions expressed in natural language and the specificities of SQL execution, creating an issue of inconsistency.

(ii) The second challenge with the cross-domain configuration of the Spider database is the presence of a significant number of out-of-domain words. Specifically, in the creation set for database systems, 35% of the terms do not appear in the training sets used in Spider. This makes it difficult to accurately generalize and generate SQL queries from natural language input.

(iii) Recent studies suggest that current domain generalization techniques are limited. Additionally, the methods utilized by existing neural semantic parsers to handle schema linking are not effective in more practical scenarios.

2.5 Objective

The aim of this research is to address the problem of converting natural language text into SQL queries. To achieve this, a new framework is proposed that utilizes pre-trained language models (PLMs) to extract relational structures. This is done through a probing approach using the Poincare distance metric. The generated relations are then used for enhancement of schema linking in graph-based parsers.

Schema linkage is a critical step in achieving domain generalization in natural language to SQL tasks. It involves correctly matching entity reference in natural language question token to the relevant column or table in the database, even in cases where the database is previously unknown to the model. This step is challenging because it requires understanding the semantics of the natural language question and the structure of the database schema simultaneously.

This thesis work will target to achieve the following objectives.

(i) To review and analyse the state of the art methods existing for semantic parsing of natural language into relational database queries.

(ii) To implement a key research papers of NL query to structured language query conversion.

(iii) To come-up with an efficient model for graph-based Text to SQL to improve the accuracy.

(iv) To develop a parameter-free probing approach that can extract schema linking information from pre-trained language models (PLMs).

(v) To compare state-of-art method and proposed method for Text to SQL problem.

3

Proposed methodology

The complete architecture and methodology are discussed in this section, along with the implementation details. The discussion will revolve around the system architecture used for the construction of this project, the methodology proposed in this paper, and the working process and implementation details of the same.

3.1 Graph-Inspired Text to SQL Model

3.1.1 Notations

In natural language question Q and the corresponding database schema $\mathcal{S} = \langle \mathcal{T}, \mathcal{C} \rangle$, the target is to generate a SQL query Y. In other words, the question $Q = \{q_1, q_2, \cdots, q_{|Q|}\}$ is a series of tokens, and the database schma comprises of tables $T = \{t_1, t_2, \cdots, t_{|\mathcal{T}|}\}$ and columns $C = \{c_1, c_2, \cdots, c_{|\mathcal{C}|}\}$. Every table t_i has various tokens $(t_{i,1}, t_{i,2}, \cdots, t_{i,|t_i|})$ and every column name $c_j^{t_i}$ in table t_i have words $\left(c_{j,1}^{t_i}, c_{j,2}^{t_i}, \cdots, c_{j,|c_j^{t_i}|}^{t_i}\right)$. Formally, we denote the input as X, where $X = \langle Q, \mathcal{S} \rangle$ and required SQL query as Y which is described as an abstract syntax tree (AST) [28] in the context free grammar of SQL. We make use of the de facto encoder decoder framework, where encoder jointly maps schema items and Natural Language questions into embeddings X and the decoder generates abstract syntax tree of the target query Y in the depth-first-search order. In our paper, we follow two representative graph-based models RAT-SQL [34] and LGESQL [36] as our base models given their performance.

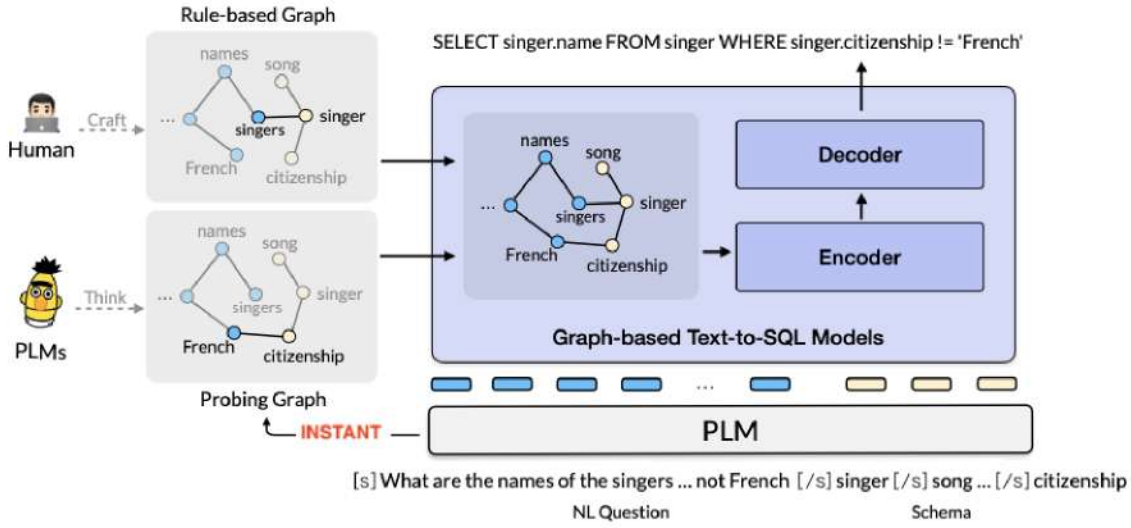

Figure 3.1: An outline of our proposed Text to SQL parsing mechanism. To get relation graphs between NL queries and database schema, we probe relational structures from PLMs using our suggested approach INSTANT. The induced relations, as well as the often used heuristic relations extracted with handmade rules, are then leveraged by a graph-based Text to SQL parser to improve schema linking for domain generalization.

3.1.2 Encoder

Conventionally, the graph-based models RAT-SQL and LGESQL consider the database schema and the Natural Language question as a single direct graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V} = Q \cup \mathcal{T} \cup \mathcal{C}$ denotes node set that contains input schema items as well as Natural Language question tokens and $\mathcal{E}$ is edge set representing preexisting relation between natural language schema item and question token. Given the inputs $X = \{x_i\}_{i=1}^{n}$ and input graph $\mathcal{G}$, a relational aware transformer (RAT) [51] works as encoder. The relation aware transformer is based on the classic transformer. Still, it representing relative position information in a self attention layer, which changes each x_i into $\boldsymbol{y}_i \in \mathbb{R}^{d_x}$:

$$e_{ij}^{(h)} = \frac{x_i W_Q^{(h)} \left(x_j W_K^{(h)} + r_{ij} \right)^\top}{\sqrt{d_z / H}} \tag{3.1}$$

$$\alpha_{ij}^{(h)} = \text{softmax}_j \left\{ e_{ij}^{(h)} \right\} \tag{3.2}$$

$$z_i^{(h)} = \sum_{j=1}^{n} \alpha_{ij}^{(h)} \left(\boldsymbol{x}_j W_V^{(h)} + r_{ij} \right) \tag{3.3}$$

$$z_i = \text{Concat} \left(\boldsymbol{z}_i^{(1)}, \cdots, \boldsymbol{z}_i^{(H)} \right) \tag{3.4}$$

$$\tilde{\boldsymbol{y}}_i = \text{LayerNorm} \left(\boldsymbol{z}_i + \boldsymbol{x}_i \right) \tag{3.5}$$

$$\boldsymbol{y_i} = \text{LayerNorm} \left(\tilde{\boldsymbol{y}}_i + \text{FC} \left(\text{ReLU} \left(\text{FC} \left(\tilde{\boldsymbol{y}}_i \right) \right) \right) \right) \tag{3.6}$$

where FC represents the Fully Connected layer operation and LayerNorm represents the Layer-Normalization [37]. The RAT framework uses the classic transformer but includes correlative position information in self-attention layer. In this layer, every input token x_i is transformed into a vector $\boldsymbol{y}_i \in \mathbb{R}^{d_x}$ using learnable parameters $W_Q^{(h)}, W_K^{(h)}, W_V^{(h)} \in$

$\mathbb{R}^{d_x \times (d_x/H)}$, where d_x is the dimension of the hidden representation, and H is the number of heads. The relationship between input elements x_i and x_j is encoded as r_{ij}, which is a learned embedding or a zero vector in case when the relations do not hold. RAT framework also includes preexisting features for every edge (i, j) as r_{ij}. More details on the implementation of RAT can be found in [34].

LGESQL uses a line-graph augmented relational-graph attention network (RGAT) as an encoder. Unlike RAT, RGAT is inspired on a graph based attention network and incorporates relative location information in a self attention layer. In addition to the original node-centric graph, LGESQL adds an extra edge-centric line graph. Every node in network merge details from its neighbourhood and includes edge attributes from the dual-graph throughout the node embedding iteration process to update its representation. The details of RGAT are not provided here, but can be found in the original paper [36].

3.1.3 Decoder

The grammar-based syntactic neural decoder used by both RAT-SQL and LGESQL is established on the work of Peng *et al.* [28] and is used in the generation of abstract syntax tree (AST) of the target SQL query Y in a depth first search order. In every step where decoding is performed, the output provided by the decoder can be either of the two actions. APPLY RULE which causes the expansion of the current non terminal node in partially created AST. SELECT TABLE or SELECT COLUMN which selects a schema item the output memory of the encoder. For more detail on how these models work, readers can refer to [34].

Discussion of Schema Linking. Referring to the above, $r_{i,j}$ in Eqn(1) and Eqn(3) illustrates the items in schema linking item in inputs. The graphs used in LGESQL and RAT-SQL are built via lexical matching and schema linking. In an NL inquiry, for example, the term "cylinders" will be connected to the column maintaining the data of cylinders in the table cars data. In this manner, a relation aware input graph, in the representation of an adjacency matrix, may be created. Nevertheless, in more difficult

cases, when implicit references are carried by the NL inquiries, such as synonym replacement or abbreviating entities, the process of string matching based on the rules cannot be applied. The encoder's potential to record significant relationships may be hampered by the missing connectivity.

We propose in this research to indirectly investigate schema connection information from large-scale pre-train language model that are supposed to have extensive semantic relation information. In the following part, we will go through specifics of our proposed probing approach.

3.2 Probing Schema Linking

Here, we present a parameter-free probing method to investigate the schema linking knowledge between database schema and the Natural language query using PLMs, as illustrated in Figure 3.2. Specifically, we introduce a masking approach to quantify the connectivity between natural language query tokens and database entities (i.e., tables and columns) during MLM process.

3.2.1 Probing Stage

Database schema $\mathcal{S} = \langle \mathcal{T}, \mathcal{C} \rangle$, here T is table and C is column and $\mathcal{T} = \left(t_1, t_2, \cdots, t_{|\mathcal{T}|}\right)$ and $\mathcal{C} = \left(c_1, c_2, \cdots, c_{|\mathcal{C}|}\right)$ respectively, we do concatenation on $\mathcal{T} and \mathcal{C}$ to single sequence $\mathcal{S} = (\mathcal{T}, \mathcal{C}) = \left(s_1, s_2, \ldots, s_{|\mathcal{T}|+|\mathcal{C}|}\right)$. This with natural language question sequence $Q = \left(q_1, q_2, \ldots, q_{|Q|}\right)$, we form the input $\mathcal{I}$ by sequentially concatenating Q and $\mathcal{S}$ as:

$$\mathcal{I} = \left(\langle \mathbf{s} \rangle; q_1; \ldots; q_{|Q|}; \langle \backslash \mathbf{s} \rangle; s_1; \langle \langle \mathbf{s} \rangle; \ldots; \langle \backslash \mathbf{s} \rangle; s|\mathcal{T}| + |\mathcal{C}|\right)$$

where $\langle s \rangle, \langle \backslash s \rangle$ are special tokens to bind the input tokens. The MLM process does the maping of input $\mathcal{I}$ into the deep contextualized representation.

The objective of our proposed probing technique is to develop a function $f(\cdot, \cdot)$ that captures relation between any pair of a schema item and a question token. To accomplish this goal, two step MLM process is used. This approach is stimulated by detecting that

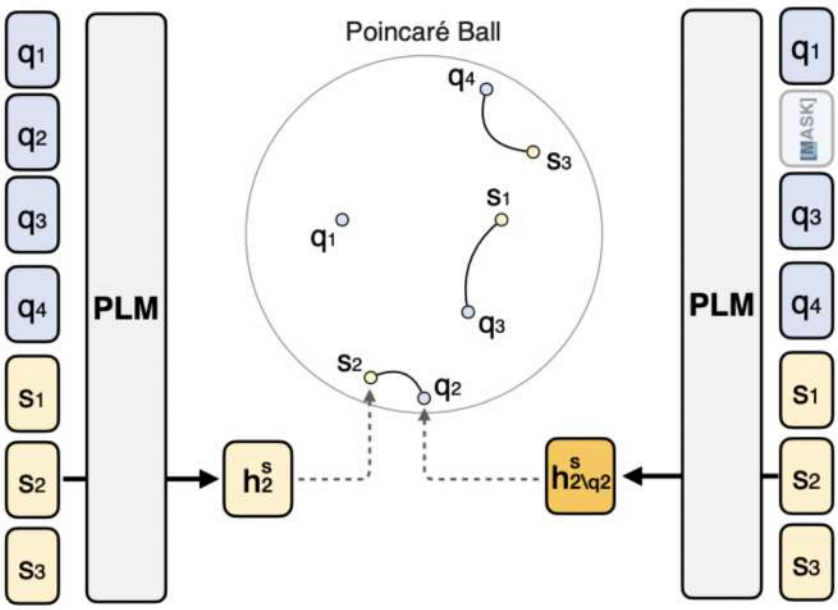

Figure 3.2: An overview of probing process for INSTANT. h_2^s discribe the embedding of schema item s_2, where as $h_{2\backslash q2}^s$ implies embedding when question token q_2 is masked out. The relation between schema items and question tokens can be seen in the Poincare ball.

a word is deemed important for document classification if after removal of it from a document results leads to a significant reduction in accuracy.

We first the input I is fed to the pre-train language model, as shown in Figure 3.2. We use h_j^s to represent the contextualized presentation of the j-th schema item s_j, where $1 \leq j \leq |\mathcal{T}| + |\mathcal{C}|$. Which is followed by replacing question token q_i with a mask token [MASK], and the modified input is fed into the pre-train language model. Here $\mathrm{h}_{j\backslash q_i}^s$ indicate new representation of the j-th schema item when q_i is masked out.
Then we measure the distance between h_j^s and $\mathrm{h}_{j\backslash q_i}^s$ to find the correlation between the database schema item s_j and the question token q_i as follows:

$$f\left(q_i, s_j\right) = d\left(\mathrm{h}_{j\backslash q_i}^s,\ \mathrm{h}_j^s\right) \tag{3.7}$$

where $d(\cdot, \cdot)$ represents distance metric to calculate the difference between two vectors. Generally, we use Euclidean distance metric to implement $d(\cdot, \cdot)$:

$$d_{\text{Euc}}\ (q_i, s_j) = \left\| \mathrm{h}^s_{j\setminus q_i} - \mathrm{h}^s_j \right\|_2 \tag{3.8}$$

where $d_{\text{Euc}}\ (\cdot, \cdot)$ represents a distance function in Euclidean space.

3.2.2 Poincare Probing

The Euclidean space is not well-suited for modeling complex data due to inherent limitations [38]. To address this issue and best capture the heterogeneous relation between database schema and the natural language query, we propose a Poincaré probe. This method utilizes hyperbolic space for schema linking information from pre train language models, which is expected to more accurately capture the linguistic hierarchies present in contextualized representations [39,40]. Previous research has demonstrated that using hyperbolic space allows for vector comparison with lower distortion than that of Euclidean space [41], and it may reflect some graph properties naturally [39,41,42].

The Poincare Ball. The standard Poincare ball is used in this paper to capture the difference between $\mathrm{h}^s_{j\setminus q_i}$ and h^s_j. The Poincare ball is a model of hyperbolic spaces, and its basic concepts are reviewed before introducing the Poincare Probe in Ganea et al. [43]. For a point x in the hyperbolic space, the Poincare ball model is represented by $\left(\mathbb{D}^n, g^{\mathbb{D}}_{\mathrm{x}}\right)$, where $\mathbb{D}^n = \{\mathrm{x} \in \mathbb{R}^n \mid \|\mathrm{x}\|^2 < 1\}$ is a Riemannian manifold and $g^{\mathbb{D}}_{\mathrm{x}} = (\lambda_{\mathrm{x}})^2\, \mathrm{I}_n$ is the metric tensor. The conformal factor is formulated as $\lambda_{\mathrm{x}} = 2/\left(1 - \|\mathrm{x}\|^2\right)$. Here, n denotes the dimension size. This standard Poincare ball model is used in the Poincare Probe to capture the hierarchical relationships between schema items and question tokens.

Feature Projection. The feature vectors acquired by pre train language models are contrasted in the hyperbolic space utilizing the exponential mapping function $g_{\mathrm{x}}(\cdot)$ that projects the embeddings into the hyperbolic space.

Suppose $h \in \mathbb{T}^n_{\mathrm{y}}$ is the input vector in the tangent space with respect to the point y in the hyperbolic space. Here, h will be instantiated as $\mathrm{h}^s_{j\setminus q_i}$ and h^s_j. The mapping function $g_{\mathrm{y}}(\cdot) : \mathbb{T}^n_{\mathrm{y}} \to \mathbb{D}^n$ can be computed by:

$$g_{\mathrm{y}}(\mathrm{h}) = \mathrm{y} \oplus \left(\tanh\left(\frac{\lambda_y \| \mathrm{h} \|}{2} \right) \frac{\mathrm{h}}{\|\mathrm{h}\|} \right) \tag{3.9}$$

where $\oplus$ is the Möbius addition. For any a, $b \in \mathbb{D}^n$, it is calculated as:

$$\mathrm{a} \oplus \mathrm{b} = \frac{(1 + 2\langle \mathrm{a}, \mathrm{b} \rangle + \|\mathrm{b}\|^2)\,\mathrm{a} + (1 - \|\mathrm{a}\|^2)\,\mathrm{b}}{1 + 2\langle \mathrm{a}, \mathrm{b} \rangle + \|\mathrm{a}\|^2 \| \mathrm{b}\|^2} \tag{3.10}$$

In our proposed work, we take the assumption that h lies in tangent space at the point $x = 0$. Then, the hyperbolic representation $\tilde{h}$ of h can be obtained by:

$$\tilde{\mathrm{h}} = g_0(\ \mathrm{h}) = \tanh(\|\mathrm{h}\|) \frac{\mathrm{h}}{\|\mathrm{h}\|} \tag{3.11}$$

Calculating Poincare Distance. When the feature representations are acquired by utilizing the feature mapping function $g_{\mathrm{x}}(\cdot)$ in the hyperbolic space, the calculation of the Poincare distance between $\tilde{\mathrm{h}}j^s$ and $\tilde{\mathrm{h}}j \backslash {q_i}^s$ allows us to compute the association between the question token q_i and the schema item s_j as given:

$$d_{\mathrm{Poin}}\ (q_i, s_j) = 2 \tanh^{-1}\left(\left\| -\tilde{\mathrm{h}}^s_{j \backslash q_i} \oplus \tilde{\mathrm{h}}^s_j \right\| \right) \tag{3.12}$$

where the Möbius addition is represented by $\oplus$ as determined in Eq. (10). The implementation of function $f(\cdot, \cdot)$ for relational knowledge probing is done by applying the Poincare distance $d_{\mathrm{Poin}}\ (\cdot, \cdot)$ rather than the Euclidean distance $d_{\mathrm{Euc}}(, , \cdot)$ as determined in Eq. (8)

3.2.3 Schema Linking for Graph Construction

When the two stage MLM operation is repeated on the pair of tokens q_i, s_j and through the calculation of $f\ (q_i, s_j)$, a relation matrix $\mathrm{X} = \{x_{i,j}\}_{i=1,j=1}^{|Q|,|S|}$ is acquired, where the relationship of the question schema pair (q_i, s_j) is denoted by $x_{i,j}$. The min max normalization is employed to decrease the influence of the correlation scores' range:

$$\tilde{x}_{i,j} = \frac{x_{i,j} - \min(\mathrm{X})}{\max(\mathrm{X}) - \min(\mathrm{X})} \tag{3.13}$$

Now, we can derive a method to construct the unweighted direct graph G used in RAT-SQL and LGESQL. To be precise, we convert the relation matrix X into an adjacency

matrix A that represents the structure of graph $\mathcal{G}$. We compute the adjacency matrix A by following:

$$\mathrm{A}_{ij} = \begin{cases} 0 & \text{if } \tilde{x}_{i,j} < \tau \\ 1 & \text{if } \tilde{x}_{i,j} > \tau \end{cases}, \tag{3.14}$$

where τ is a pre-defined threshold. The learned adjacency matrix A via probing the relational knowledge from PLMs can facilitate the semantic linking of Text to SQL parsing.

3.3 Experimantal Setup

In this section, we will discuss about dataset along with baselines and implementation details.

3.3.1 Datasets

The proposed approach will be evaluated through extensive experiments on three benchmark datasets. These datasets have been widely used in the Text to SQL parsing domain and have different levels of complexity: (1) Spider [23] is a cross-domain, large zero shot Text to SQL benchmark. Due to unavailability of the test set, we provide precise match accuracy for the development set. (2) DK [44] dataset is a challenging variation of the Spider development set that focuses on evaluating the model's understanding of domain knowledge. It is a human-curated dataset used to assess models' performance in handling complex schema structures and domain-specific terminologies. (3) SYN [45] is another challenging variation of Spider. It was created by manually modifying natural language questions in Spider by substituting synonym. It aims to create a scenario where users don't know the exact schema words in the observation.

3.3.2 Baselines

We have chosen two base parsers, RAT-SQL and LGESQL, as benchmarks for our model. RAT-SQL is a sequence to sequence model that incorporates a relational aware transformer, while LGESQL is a graph-attention network based sequence to sequence model

that uses the relational GAT and line graph. Both these models uses schema linking to create input graph. Additionally, to ensure a thorough comparison, we have also evaluated our model against several recent models, including IRNet [32], EditSQL [33],GNN [24], TranX [30], RYANSQL [27]. For RATSQL, we choose RAT-SQL + Grappa as our baseline, which has great performance of RAT-SQL [31].

3.3.3 Implementation Details

RAT-SQL's encoder architecture includes:

- Bidirectional LSTMs with a hidden size of 128 in each direction.
- 8 relation aware self attention layers on top of the bi-directional LSTMs, with a dimension of 256 for each attention layer.
- A position-wise feed-forward network with an inner layer dimension of 1024.

In the decoder:

- The rule embedding has a dimension of 128.
- The node type embedding has a dimension of 64.
- The hidden state has a dimension of 512.
- The model uses the Adam optimizer with default hyperparameters and trains with a batch size of 8 and 40,000 training steps.

The model uses the Adam optimizer [46] with default hyperparameters and trains with a batch size of 8 and 40,000 training steps.
LGESQL's architecture includes:

- LGESQL uses a GNN hidden size of 512 for PLMs in the encoder.
- It includes 8 GNN layers in the encoder.

- The decoder has hidden state, action embedding, and node type embedding dimensions of 512, 128, and 128, respectively.
- The decoder LSTM has a recurrent dropout rate of 0.2.
- Multi-head attention uses 8 heads.
- Dropout rate of features is 0.2 in both encoder and decoder.
- AdamW optimizer [47] is used with a linear warmup scheduler and a warmup ratio of 0.1 for total training steps
- Maximum gradient norm is set to 5 with batch size of 20.

4

Results

This chapter talks about the various experiments performed over the duration of this thesis and results achieved from the experiments performed.

4.1 Results

Table 1 presents the results obtained on the DK dataset. Based on the results, the following observations can be made. The results show that LGESQL performs better than the considered baseline methods on all three datasets when using the Euclidean INSTANT probing method. Specifically, the LGESQL+ELECTRA-large model with Euclidean INSTANT achieves an exact matching accuracy of 49.3% on the DK benchmark, which is a 0.9% improvement over LGESQL+ELECTRA-large. The results demonstrate that utilizing semantic and relational knowledge from pre-trained language models can improve the ability of Text to SQL parsers to comprehend domain knowledge and generalize to new domains. This trend is also evident in the SYN benchmark and the Spider dataset.

Table 4.1: Accuracy comparisons of the proposed method with the various state-of-the-art existing works. Exact matching accuracy on DK benchmark.

Model	Accuracy
GNN + BERT [24]	26.0
IRNet + BERT [32]	33.1
RAT-SQL [34]	35.8
RAT-SQL + BERT [34]	40.9
RAT-SQL + GAP [29]	44.1
RAT-SQL + Grappa	38.5
w/ Euclidean INSTANT	42.0(↑ 3.5)
w/ Hyperbolic INSTANT	**44.4**(↑ 5.9)
LGESQL + RoBERTa-large	45.9
w/ Euclidean INSTANT	46.2(↑ 0.3)
w/ Hyperbolic INSTANT	**46.7**(↑ 0.8)
LGESQL + ELECTRA-large	48.4
w/ Euclidean INSTANT	49.3(↑ 0.9)
w/ Hyperbolic INSTANT	**51.0**(↑ 2.6)

Table 4.2: Accuracy comparisons of the proposed method with the various state-of-the-art existing works. Exact matching accuracy on SYN benchmark.

Model	Accuracy
GNN [24]	23.6
IRNet [32]	28.4
RAT-SQL [34]	33.6
RAT-SQL + BERT [34]	48.2
RAT-SQL + Grappa	49.1
w/ Euclidean INSTANT	53.4(↑ 4.3)
w/ Hyperbolic INSTANT	**54.6**(↑ 5.5)
LGESQL + RoBERTa-large	54.1
w/ Euclidean INSTANT	57.7(↑ 1.6)
w/ Hyperbolic INSTANT	**56.6**(↑ 2.5)
LGESQL + ELECTRA-large	64.6
w/ Euclidean INSTANT	65.4(↑ 0.8)
w/ Hyperbolic INSTANT	**65.5**(↑ 0.9)

Table 4.3: Accuracy comparisons of the proposed method with the various state-of-the-art existing works. Exact matching accuracy on Spider dev set.

Model	Accuracy
IRNet + BERT [32]	61.9
RYANSQL + BERT [27]	70.6
TranX + TaBERT [30]	65.2
RAT-SQL + BERT [34]	69.7
RAT-SQL + Grappa	71.2
w/ Euclidean INSTANT	71.6(↑ 0.4)
w/ Hyperbolic INSTANT	**72.1**(↑ **0.9**)
LGESQL + RoBERTa-large	71.7
w/ Euclidean INSTANT	72.9(↑ 1.2)
w/ Hyperbolic INSTANT	**73.3**(↑ 1.6)
LGESQL + ELECTRA-large	75.3
w/ Euclidean INSTANT	76.0(↑ 0.7)
w/ Hyperbolic INSTANT	**76.1**(↑ 0.8)

4.2 Schema Linking Performance Analysis

In order to obtain a more thorough understanding of how INSTANT assists in acquiring relational knowledge from PLMs, we meticulously perform a process of error checking related to schema linking on the Spider benchmark. We perform analysis on errors made

by previous works [34, 36] and classifying them into four categories: World Knowledge Error, Type Error , Semantic Understanding Error and Inference Required Error. We have noticed that INSTANT is capable of resolving the majority of bad cases that previous techniques were unable to handle.

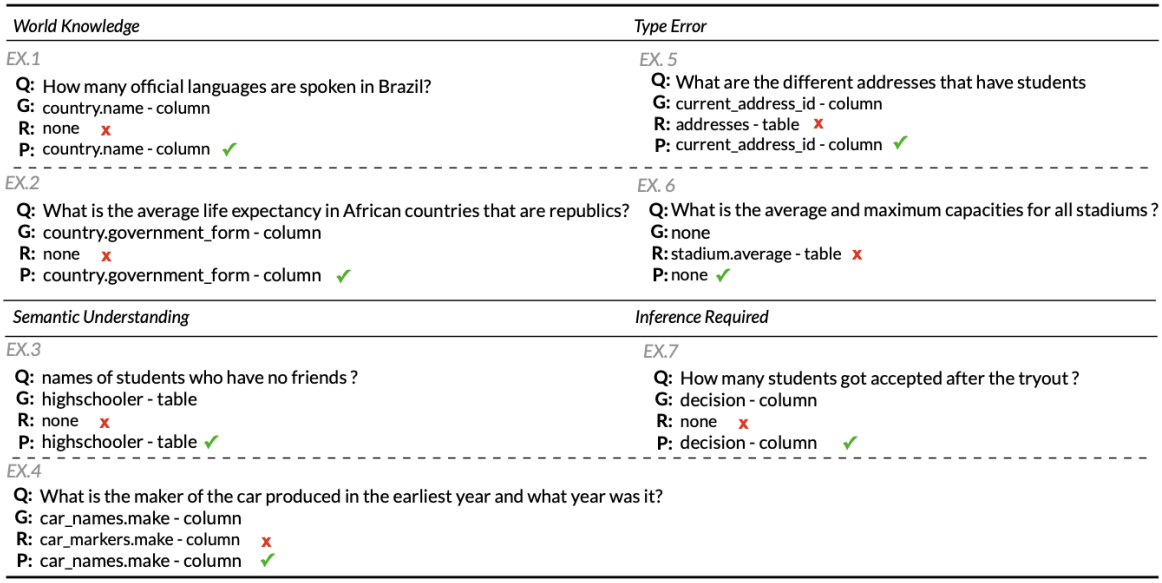

Figure 4.1: Typical inaccurate predictions regarding schema linking made by rule-based methods. Notations Q, G, R, and P stand for question, ground truth, rule-based approach prediction and INSTANT prediction, respectively.

Initially, we have discovered that a majority of incorrect predictions stem from a deficiency in worldly knowledge. In the first illustration depicted in Figure 4, rule-based semantic linking with exact text matching fails to recognize"Brazil" as a country name and also misinterprets "republic" as a type of government in the second example. Although pre-training PLMs on large-scale corpora could potentially offer external knowledge, previous models employing PLMs are still unable to grasp reference linking [34, 36]. This limitation arises from using only PLMs embeddings which are not effective in eliciting relational knowledge. On the other hand, our probing method is capable of eliciting such worldly (relational) knowledge, and is feasible for practical use.

Second, one of the error categories is caused by an inability to capture semantic relations between words and tables/columns. The rule-based schema linking tends to select

tables/columns that precisely appear in the query through exact string matching. In the fourth example in Figure 4, the rule-based schema linking connects the term "maker" to the column "car_makers.Maker", whereas the correct selection should be the column "car_names.Make". Similarly, the third example fails to acknowledge that "highschooler" is a synonym for "students". This situation can be resolved by considering in-depth semantic information of the query rather than just word occurrence. Our probing method can conveniently solve this type of problem.

Furthermore, in some instances, questions may have more than one entity word that corresponds to table names. The rule-based schema linking could erroneously select columns that match the entity names present in the query. For instance, in the fifth example, the rule-based schema linking incorrectly predicts the column "Address" instead of the correct column "current_address_id". Moreover, in specific cases, when the column name precisely matches the keyword in the question, the rule-based schema linking struggles to differentiate it clearly, as demonstrated in the sixth example. This issue frequently arises when the question contains words like "average," "maximum," and "minimum," etc.

Finally, the rule-based schema linking is unable to address challenging cases that require strong inference ability. In the seventh example, for instance, the term "accepted" in the question implicates the column "decision," which cannot be identified through exact string matching. Our INSTANT model can resolve these difficult cases by utilizing semantic knowledge obtained from PLMs.

Question	Find the type and height of the youngest pet.
SYN_Question	Find the category and height of the youngest pet.
LGESOL	SELECT Pets.Pet_age, Pets.height FROM Pets ORDER BY Pets.pet_age LIMIT 1
INSTANT	SELECT Pets.PetType, Pets.height FROM Pets ORDER BY Pets.pet_age LIMIT 1
Question	How many distinct countries do players come from?
SYN_Question	How many distinct states do participants come from
LGESOL	SELECT COUNT(*) FROM matches
INSTANT	SELECT COUNT(DISTINCT players.country_code) FROM players
Question	What are the names of the singers who are not German?
LGESQL	SELECT singer.Name FROM singer WHERE singer. Name != 'German'
INSTANT	SELECT singer Name FROM singer WHERE singer.Citizenship != 'German'
Question	Find the average and maximum id for each type of pet.
LGESQL	SELECT Pets.PetType, Avg(Pets.PetType), Max(Pets.PetType) FROM Pets GROUP BY Pets.PetType
INSTANT	SELECT Pets. PetType, Avg(Pets.PetID), Max(Pets.PetID) FROM Pets GROUP BY Pets.PetType

Figure 4.2: Case Study:The first two cases are chosen from the SYN dataset, while the last two cases are selected from the DK dataset.

4.3 Case Study

To showcase the effectiveness of our proposed model, we selected four representative cases from DK and SYN benchmarks and present them in Table 4.2, where the top two cases are from SYN and the bottom two cases are from DK. The table compares the generated SQL queries of LGESQL, the best baseline, and our Instant model. The results demonstrate that our model outperforms the baseline in challenging scenarios like synonym substitution. In the first case, for example, when the word "type" in the NL question is replaced with its synonym "category," LGESQL fails to identify the correct column PetType, whereas our Instant model generates the correct SQL query.

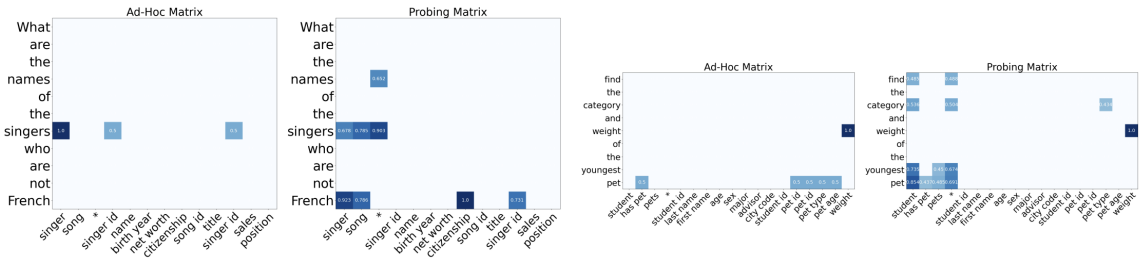

Figure 4.3: The Rule-based Matrix and Probing Matrix are visualized for two cases, with the left two sub-figures corresponding to the first case (SYN) in Table 4.4, and the right two sub-figures corresponding to the third case (DK) in Table 4.4

Figure 4.3 depicts the visualization of correlation matrices obtained through both rule-

based feature engineering and our probing technique. The two sub-figures on the right illustrate that the rule-based schema linking approach, which relies on exact string matching, fails to recognize the relationship between the term "category" and the corresponding table column named PetType. However, with the aid of knowledge extracted from large-scale pre-train language models, Instant can easily detect this semantic similarity and generate the correct SQL query. Similarly, in the two sub-figures on the left, Instant successfully links the domain knowledge term "French" to the column named Citizenship, while LGESQL struggles to identify the column without explicit mentions. In our opinion, Instant can effectively extract rich semantic and relational knowledge from large-scale PLMs, which can improve the accuracy of schema linking in Text to SQL parsing.

5

Discussions and Conclusions

In this chapter, the work is concluded and future plan is presented. Next, the limitation of work and possible future extensions are described respectively.

5.1 Conclusions

Chapter 2 provides a comprehensive review of research papers, highlighting their unique characteristics, advantages, and disadvantages. These studies offer valuable insights into Text to SQL parsing, but also suggest areas for future research and development. Upon examining the review, it can be inferred that standard benchmarks have limitations in evaluating domain generalization, and the current neural semantic parsers methods for handling schema linking are not effective in generalizing to more practical scenarios. The paper proposes a probing method to extract schema-linking information from large-scale pre-train language models, which can improve the domain generalization and strength of Text to SQL parsing models. Additionally, a Poincaré distance metric is developed to capture the heterogeneous relational structures between the NL query and the database schema by measuring the distance between two vectors in the hyperbolic space. The experiments conducted on three benchmark datasets showed that our approach outperformed strong baseline models on all three Text to SQL benchmarks.

5.2 Limitations

Limitations of this work include the fact that the evaluation was performed only on the Spider dataset, which is currently the only open-domain large dataset containing complex SQL queries in the text-to-SQL domain. In order to better reflect real-world scenarios, additional datasets with SQL queries that more closely resemble those encountered in practice are needed. Additionally, there are still a lot of cases where conditional values are predicted wrongly. Also some queries that needed a JOIN or required a GROUP BY clause showed wrong output.

5.3 Future Scope

In the future, it is possible to extend this work to address the limitations mentioned above using more efficient methods. As more datasets with real-world SQL queries become

available, our model can be re-evaluated on these datasets to demonstrate its effectiveness in cross-domain scenarios. Additionally, future research can focus on developing more efficient ways to encode and decode the characteristics of natural language sentences for improved performance.

Bibliography

[1] R. Churches and R. Terry, *NLP for teachers: How to be a highly effective teacher.* Crown House Publishing, 2007.

[2] S. F. Khan and M. A. A. Raheed, *A Developer's Guide To Database Management Systems: Using Oracle 10g RDBMS.* CreateSpace Independent Publishing Platform, 2015.

[3] L. Ullman, *Php and mysql for dynamic web sites: visual quickpro guide.* Peachpit Press, 2011.

[4] S. Iyer, I. Konstas, A. Cheung, J. Krishnamurthy, and L. Zettlemoyer, "Learning a neural semantic parser from user feedback," *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 963–973, 2017.

[5] F. Li and H. V. Jagadish, "Constructing an interactive natural language interface for relational databases," *Proceedings of the VLDB Endowment*, vol. 8, pp. 73–84, 2014.

[6] A. Giordani and A. Moschitti, "Translating questions to sql queries with generative parsers discriminatively reranked," *Proceedings of COLING 2012: Posters*, vol. 1, pp. 401–410, 2012.

[7] K. Xu, L. Wu, Z. Wang, Y. Feng, and V. Sheinin, "SQL-to-text generation with graph-to-sequence model," *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, vol. 2.

[8] P. Pasupat and P. Liang, "Compositional semantic parsing on semi-structured tables," *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, vol. 1, pp. 1470–1480, 2015.

[9] T. Yu, M. Yasunaga, K. Yang, R. Zhang, D. Wang, Z. Li, and D. Radev, "SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task," *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, vol. 2, pp. 1653–1663, 2018.

[10] N. Yaghmazadeh, Y. Wang, I. Dillig, and T. Dillig, "Sqlizer: query synthesis from natural language," *Proceedings of the ACM on Programming Languages*, vol. 1, pp. 63–89, 2017.

[11] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, vol. 4, pp. 1533–1544, 2013.

[12] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, vol. 7, pp. 55–60, 2014.

[13] C. Liu, X. Chen, R. Shin, M. Chen, and D. Song, "Latent attention for if-then program synthesis," *Advances in Neural Information Processing Systems*, vol. 29, pp. 4574–4582, 2016.

[14] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[15] Y. Li, H. Yang, and H. Jagadish, "Constructing a generic natural language interface for an xml database," *International Conference on Extending Database Technology*, vol. 3896, pp. 737–754, 2006.

[16] T. Yu, Z. Li, Z. Zhang, R. Zhang, and D. Radev, "TypeSQL: Knowledge-based type-aware neural text-to-SQL generation," *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, vol. 1, pp. 588–594, 2018.

[17] S. Reddy, M. Lapata, and M. Steedman, "Large-scale semantic parsing without question-answer pairs," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 377–392, 2014.

[18] P. Yin, Z. Lu, H. Li, and K. Ben, "Neural enquirer: Learning to query tables in natural language," *Proceedings of the Workshop on Human-Computer Question Answering*, vol. 2, pp. 29–35, 2016.

[19] L. Dong and M. Lapata, "Language to logical form with neural attention," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 33–43, 2016.

[20] M. Song, Z. Zhan, and E. Haihong, "Hierarchical schema representation for text-to-sql parsing with decomposing decoding," *IEEE Access*, vol. 7, pp. 103 706–103 715, 2019.

[21] R. Cai, B. Xu, Z. Zhang, X. Yang, Z. Li, and Z. Liang, "An encoder-decoder framework translating natural language to database queries," *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, vol. 18, p. 3977–3983, 2018.

[22] L. Zettlemoyer and M. Collins, "Online learning of relaxed ccg grammars for parsing to logical form," *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, vol. 1, pp. 678–687, 2007.

[23] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," *Association for Computational Linguistics*, vol. 4, pp. 3911–3921, 2018.

[24] J. B. Ben Bogin and M. Gardner, "Representing schema structure with graph neural networks for text-to-sql parsing." *ACL*, 2019.

[25] Q. R. B. L. Y. L. J. S. F. H. L. S. P. Z. Binyuan Hui, Ruiying Geng and X. Zhu., "Dynamic hybrid relation exploration network for cross-domain context-dependent semantic parsing." *AAAI*, 2021.

[26] L. W. B. Q. B. L. J. S. Binyuan Hui, Ruiying Geng and Y. Li, "Sql: Injecting syntax to question-schema interaction graph encoder for text-to-sql parsers." *ACL*, 2022.

[27] E. K. Donghyun Choi, M. Shin and D. R. Shin, "Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases." *ArXiv abs/2004.03125*, 2020.

[28] P. Yin and G. Neubig, "A syntactic neural model for generalpurpose code generation." *ACL*, 2017.

[29] Z. g. W. H. Z. A. H. L. J. W. C. D. S. Peng Shi, Patrick Ng and B. Xiang, "Learning contextual representations for semantic parsing with generation-augmented pre-training," *AAAI*, 2021.

[30] W. t. Y. Pengcheng Yin, Graham Neubig and S. Riedel, "Tabert: Pretraining for joint understanding of textual and tabular data." *ACL*, 2020.

[31] X. V. L. B. W. Y. T. X. Y. D. R. R. S. Tao Yu, Chien-Sheng Wu and C. Xiong, "Grappa: Grammaraugmented pre-training for table semantic parsing." *ICLR*, 2021.

[32] Y. G. Y. X. J.-G. L. T. L. Jiaqi Guo, Zecheng Zhan and D. Zhang, "Towards complex text-to-sql in cross-domain database with intermediate representation," *ACL*, 2019.

[33] Y. E. H. S. S.-X. E. V. L. X. S. T. X. C. S. R. Zhang Rui, Yu Tao and R. Dragomir, "Editingbased sql query generation for cross-domain context-dependent question." *EMNLP*, 2019.

[34] X. L. O. P. Bailin Wang, Richard Shin and M. Richardson, "Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers." *ACL*, 2020.

[35] N. S. Torsten Scholak and D. Bahdanau., "Parsing incrementally for constrained auto-regressive decoding from language," *EMNLP*, 2021.

[36] Z. C. Y. Z. S. Z. Ruisheng Cao, Lu Chen and K. Yu., "Lgesql: Line graph enhanced text-to-sql model with mixed local and nonlocal relations," *ACL*, 2021.

[37] J. R. K. Jimmy Lei Ba and G. E. Hinton, "Layer normalization," *arXiv preprint*, 2016.

[38] Y. L. A. S. Michael M Bronstein, Joan Bruna and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data." *IEEE Signal Processing Magazine*, 2017.

[39] M. Nickel and D. Kiela., "Poincaré embeddings for learning hierarchical representations." *NeurIPS*, 2017.

[40] G. B. Alexandru Tifrea and O.-E. Ganea., "Poincare glove: Hyperbolic word embeddings." *ICLR*, 2019.

[41] C. R. Ines Chami, Zhitao Ying and J. Leskovec., "Hyperbolic graph convolutional neural networks," *NeurIPS*, 2019.

[42] M. K. A. V. Dmitri Krioukov, Fragkiskos Papadopoulos and M. Boguná., "Hyperbolic geometry of complex networks," *Physical Review E*, 2010.

[43] G. B. Octavian-Eugen Ganea and T. Hofmann., "Hyperbolic neural networks." *NeurIPS*, 2018.

[44] X. C. Yujian Gan and M. Purver, "Exploring underexplored limitations of cross-domain text-to-sql generalization," *EMNLP*, 2021.

[45] Q. H. M. P. J. R. W. J. X. Yujian Gan, Xinyun Chen and P. Huang, "Towards robustness of text-to-sql models against synonym substitution," *ACL*, 2021.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[47] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization." *arXiv preprint arXiv:1711.05101*, 2017.

khatri2

ORIGINALITY REPORT

17%	12%	11%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to ABV-Indian Institute of Information Technology and Management Gwalior Student Paper	6%
2	dspace.mit.edu Internet Source	5%
3	deepai.org Internet Source	3%
4	Submitted to HELP UNIVERSITY Student Paper	1%
5	Lihan Wang, Bowen Qin, Binyuan Hui, Bowen Li, Min Yang, Bailin Wang, Binhua Li, Jian Sun, Fei Huang, Luo Si, Yongbin Li. "Proton: Probing Schema Linking Information from Pre-trained Language Models for Text-to-SQL Parsing", Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022 Publication	1%
6	Submitted to Oxford & Cherwell Valley College Student Paper	1%