Proton: Probing Schema Linking Information from Pre-trained Language Models for Text-to-SQL Parsing

Lihan Wang†
Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
Shenzhen, China
lh.wang1@siat.ac.cn

Bowen Qin†
Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
Shenzhen, China
bw.qin@siat.ac.cn

Binyuan Hui∗
Alibaba Group
Beijing, China
binyuan.hby@alibaba-inc.com

Bowen Li
Alibaba Group
Beijing, China
yanjin.lbw@alibaba-inc.com

Min Yang‡
Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
Shenzhen, China
min.yang@siat.ac.cn

Bailin Wang
Massachusetts Institute of Technology
Cambridge, United States
bailinw@mit.edu

Binhua Li
Jian Sun
Alibaba Group
Beijing, China

Fei Huang
Luo Si
Alibaba Group
Beijing, China

Yongbin Li‡
Alibaba Group
Beijing, China
shuide.lyb@alibaba-inc.com

abstract
The importance of building text-to-SQL parsers which can be applied to new databases has long been acknowledged, and a critical step to achieve this goal is schema linking, i.e., properly recognizing mentions of unseen columns or tables when generating SQLs. In this work, we propose a novel framework to elicit relational structures from large-scale pre-trained language models (PLMs) via a probing procedure based on Poincaré distance metric, and use the induced relations to augment current graph-based parsers for better schema linking. Compared with commonly-used rule-based methods for schema linking, we found that probing relations can robustly capture semantic correspondences, even when surface forms of mentions and entities differ. Moreover, our probing procedure is entirely unsupervised and requires no additional parameters. Extensive experiments show that our framework sets new state-of-the-art performance on three benchmarks. We empirically verify that our probing procedure can indeed find desired relational structures through qualitative analysis. Our code can be found at https://github.com/AlibabaResearch/DAMO-ConvAI.

CCS Concepts
- Computing methodologies → Natural language processing;
- Discourse, dialogue and pragmatics;

Keywords
Semantic parsing, text-to-SQL parsing, knowledge probing

1 Introduction
Text-to-SQL parsing aims at converting a natural language (NL) question to its corresponding structured query language (SQL) in the context of a relational database. Although relational databases can be efficiently accessed by skilled professionals via handcrafted SQLs, a natural language interface, whose core component relies on text-to-SQL parsing, would allow ubiquitous relational data to be accessible for a wider range of non-technical users. Hence, text-to-SQL parsing has attracted remarkable attention in both academic and industrial communities.

One challenging goal of text-to-SQL parsing is achieving domain generalization, i.e., building parsers which can be successfully applied to new domains (or databases). The availability of benchmarks [57, 58] has led to numerous advances in developing parsers with domain generalization [e.g., 5, 20, 31, 51]. Central to achieving
domain generalization is schema linking – correctly aligning entity references in the NL question to the intended schema columns or tables, even on unseen databases. As shown in Figure 1, a parser needs to link mentions singers and musicians to their corresponding column singer. The importance of schema linking to domain generalization has also been verified in [31, 50].

Recent work suggests that standard benchmarks are limited in assessing domain generalization, and methods incorporated by current neural semantic parsers to handle schema linking cannot generalize well to more realistic settings. Specifically, for the standard benchmark Spider [57], most entity mentions can be extracted by a heuristic such as string matching. For example, the mention singers in the first question in Figure 1 can be trivially linked to the column singer based on their surface forms. Many parsers [20, 51] exploit such artifacts (or shortcuts), but their good performance on Spider does not transfer to real-life settings where mentions and columns/tables are very likely to share different surface forms. For example, when the mention singer is replaced with its synonym musicians, the state-of-the-art parser LGESQL [5] fails to handle the schema-linking relations.

In this work, we propose a novel approach to elicit relational structures for schema linking from large-scale pre-trained language models (PLMs) through a probing procedure. In addition to simply encoding NL question and schema in continuous vector space using PLMs, as most previous semantic parsers do, we propose to distill discrete relational structures from PLMs. Such relational structures are extracted in an unsupervised manner, and they can be directly exploited for schema linking when generating structured programs of SQLs. We capitalize on the intuition that although relational information is already contained in the continuous representations of PLMs, neural parsers lack an optimal mechanism to benefit from such information. The algorithmic inductive biases introduced by our probing procedure would allow the underlying relational structures to be explicitly and easily exploited by neural parsers. Previous work has shown that PLMs such as BERT [15], RoBERTa [37], ELECTRA [11] store linguistic knowledge [23], world knowledge [43] and relational knowledge [41]. To our best knowledge, we are the first work to adopt probing methods to exploit relational information from PLMs for the complex structured prediction task of text-to-SQL parsing.

The relational structures extracted from PLMs hold several appealing properties which make them suitable for domain generalization of text-to-SQL parsing. First, they are domain-invariant, and this is inherited from that PLMs are usually obtained via self-supervised training on various domains of textual data. Second, they can better capture semantic correspondences than current heuristics such as n-gram matching [16, 51]. Third, they are relatively more robust to the cross-database setting. As they are elicited in an unsupervised probing procedure and not induced during in-domain training, they will not suffer from overfitting to observed training databases.

In this work, we propose a novel framework, called PROTON, which first probes the underlying relational schema-linking structures between a NL query and its database schema from a pre-trained language model, and then effectively injects it into the downstream text-to-SQL parsing models. To better model the heterogeneous relational structures, we introduce a probing procedure based on Poincaré distance metric instead of the traditional Euclidean distance metric, inspired by Chen et al. [7]. Our probing procedure is entirely unsupervised and does not require additional parameters. We empirically show the effectiveness of PROTON on several text-to-SQL benchmarks, i.e., Spider [57], SYN [16] and DK [17], and through qualitative analysis, we verify that our probing procedure can indeed find desired relational structures.

The contributions of this work can be summarized as follows:

- To boost domain generalization for text-to-SQL parsing, we propose a novel framework that utilizes relational schema-linking structures that are extracted from a PLM via an unsupervised probing process.
- To better capture the heterogeneous relational structures between NL queries and database schema, we introduce a Poincaré distance metric that can better measure semantic relevance than the typical Euclidean distance metric.
- Extensive experiments on three text-to-SQL benchmarks show that our probing method can lead to significantly better results when compared with current state-of-the-art parsers.

Though we only focus on text-to-SQL parsing in this work, we believe that the general methodology of probing underlying discrete relational structures from PLMs can be extended to related tasks that require complex reasoning over structured knowledge, such as knowledge-based question answering [19] and dialog [13, 14, 33], table-based fact checking [8] and structured data record to text generation [39].

2 GRAPH-BASED TEXT-TO-SQL MODELS

2.1 Notation Definition

Given a natural language question $Q$ and the corresponding database schema $S = (T, C)$, the target is to generate a SQL query $Y$. More specifically, the question $Q = \{q_1, q_2, \ldots, q|Q|\}$ is a sequence of tokens, and the schema consists of tables $T = \{t_1, t_2, \ldots, t|T|\}$ and columns $C = \{e_1, e_2, \ldots, e|C|\}$. Each table $t_i$ contains multiple

---

1Such artifacts might result from a biased annotation process: annotators were shown the database schema and asked to formulate queries.

2PROTON: Schema Linking Information from Pre-trainNed Language Models
words \((t_1, t_2, \ldots, t_L)\) and each column name \(c_j^t\) in table \(t_i\) contains words \((c_j^1, c_j^2, \ldots, c_j^L)\). Formally, we denote the input as \(X\), where \(X = (Q, S)\) and the desired SQL query as \(Y\) which is represented as an abstract syntax tree (AST) [53] in the context-free grammar of SQL. We employ the de facto encoder-decoder framework, where the encoder jointly maps NL questions and schema items into embeddings \(X\) and the decoder generates the AST of the target query \(Y\) in the depth-first-search order. In this paper, we adopt two representative graph-based models RAT-SQL [51] and LGESQL [5] as our base models given their SOTA performance.

### 2.2 Encoder

Formally, the graph-based models RAT-SQL and LGESQL consider the NL question and the database schema as a single direct graph \(G = (V, E)\), where \(V = Q \cup T \cup C\) denotes the node set containing the input NL question tokens as well as schema items and \(E\) is the edge set depicting pre-existing relations between NL question tokens and schema items. Given the inputs \(X = \{x_i\}_{i=1}^n\) and input graph \(G\), a relational-aware transformer (RAT) [51] is leveraged as the encoder. The relation-aware transformer is based on the classic transformer but represents relative position information in a self-attention layer, which transforms each \(x_i\) into \(y_i \in \mathbb{R}^{d_e}\) as follows:

\[
e_i^{(h)} = x_i W_Q^{(h)} (x_j W_K^{(h)} + r_{ij})^T \sqrt{d_e/H} \tag{1}
\]

\[
d_{ij}^{(h)} = \text{softmax} \left( e_{ij}^{(h)} \right) \tag{2}
\]

\[
z_i = \frac{1}{M} \sum_{j=1}^{M} d_{ij}^{(h)} (x_j W_V^{(h)} + r_{ij}) \tag{3}
\]

\[
z_i = \text{Concat}\left(z_i^{(1)}, \ldots, z_i^{(H)}\right) \tag{4}
\]

\[
y_i = \text{LayerNorm}\left(y_i + \text{FC}\left(\text{ReLU}\left(\text{FC}\left(y_i\right)\right)\right)\right) \tag{5}
\]

where FC is a fully-connected layer and LayerNorm is the layer normalization operation [1]. \(W_Q^{(h)}, W_K^{(h)}, W_V^{(h)} \in \mathbb{R}^{d_e \times (d_e/H)}\) are learnable parameters where \(d_e\) denotes the dimension of hidden representation. \(d_x\) and \(d_z\) represent the dimension of \(x\) and \(z\). \(H\) is the number of heads and we have \(1 \leq h \leq H\). Here, the term \(r_{ij}\) encodes the known relationship between the two elements \(x_i\) and \(x_j\) in the input. The RAT framework represents all the pre-existing features for each edge \((i, j)\) as \(r_{ij}\) in which each element is either a learnable embedding for each corresponding edge or a zero vector if the relation does not hold for the edge. The reader can refer to [51] for the implementation details of RAT.

LGESQL applies a line-graph enhanced relational graph attention network (RGAT) as its encoder. Different from RAT, RGAT is based on graph attention network and represents relative position information in a self-attention layer. Compared with normal RGAT, line-graph enhanced RGAT employs an additional edge-centric line graph constructed from the original node-centric graph. During the iteration process of node embeddings, each node in either graph integrates information from its neighborhood and incorporates edge features from the dual graph to update its representation. Due to the limited space, we omit the formal definition of RGAT. The reader can refer to [5] for the implementation details of LGESQL.

### 2.3 Decoder

Both RAT-SQL and LGESQL apply grammar-based syntactic neural decoder [53] to generate the abstract syntax tree (AST) of the target query \(Y\) in a depth-first-search order. The output at each decoding time step is either 1) an APPLYRULE action that expands the current non-terminal node in the partially generated AST; or 2) SELECTTABLE or SELECTCOLUMN action that chooses one schema item from the output memory of encoder. The readers can refer to [51] for more details.

**Discussion of Schema Linking.** As mentioned above, \(r_{ij}\) in Eq.1 and Eq.3 represents the schema linking items in inputs. The graphs adopted in RAT-SQL and LGESQL are constructed by schema

---

**Figure 2:** The overview of our proposed framework for text-to-SQL parsing. To obtain relation graphs among NL questions and database schema, we first use our proposed method Proton to probe relational structures from PLMs. The induced relations, along with the commonly-used heuristic relations extracted with handcrafted rules, are then utilized by a graph-based text-to-SQL parser to boost its schema linking for domain generalization.
linking with lexical matching. For instance, the word “cylinders” in a NL question will be linked to the cylinder's column in a table cars_data. In this way, a relation-aware input graph can be constructed, which is represented as an adjacency matrix. However, the rule-based string matching is inapplicable in more challenging scenarios where the NL questions contain implicit mentions such as synonym substitution and entity abbreviation. The missing linkage hindered the encoder's ability to capture salient relations. In this paper, we propose to probe schema linking information from large-scale PLMs that are claimed to contain rich semantic relational knowledge implicitly. It is noteworthy that our probing technique is model-agnostic and potentially applicable for any text-to-SQL parsing models. In the next section, we will introduce the details of our probing technique.

3 PROBING SCHEMA LINKING

In this section, we introduce a parameter-free probing technique, as illustrated in Figure 3, to probe schema linking information between the NL query and the database schema from PLMs. Concretely, we propose a masking technique to measure the correlation between NL question tokens and schema items (i.e., columns and tables) in the masked language modeling (MLM) process.

3.1 Probing Stage

Given a database schema $S = (T, C)$, where the table and column sequences are $T = \{t_1, t_2, \ldots, t_{|T|}\}$ and $C = \{c_1, c_2, \ldots, c_{|C|}\}$ respectively. We first concatenate $T$ and $C$ into a single sequence $\bar{S} = (T, C) = \{s_1, s_2, \ldots, s_{|T|+|C|}\}$. Together with the NL question sequence $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$, the input $I$ is formed by a sequential concatenation of $Q$ and $\bar{S}$ as:

$$I = (\langle s \rangle; q_1; \ldots; q_{|Q|}; \langle \langle s \rangle \rangle; s_1; \langle \langle s \rangle \rangle; \ldots; \langle \langle s \rangle \rangle; s_{|T|+|C|})$$

where $\langle s \rangle$, $\langle \langle s \rangle \rangle$ are special tokens to delimit the input tokens. The MLM maps the input $I$ into the deep contextualized representations.

The goal of our probing technique is to derive a function $f(\cdot, \cdot)$ that captures the correlation between an arbitrary pair of a question token and a schema item. To this end, we employ a two-step MLM process. It is inspired by the observation that a word is considered as essential for document classification if removing the word from a document leads to a considerable accuracy decrease.

As shown in Figure 3, we first feed the input $I$ into the PLM. We use $h_i^q$ to denote the contextualized representation of the $j$-th schema item $s_j$, where $1 \leq j \leq |T| + |C|$. Then, we replace the question token $q_i$ with a special mask token [MASK] and feed the corrupted input $I \setminus \{q_i\}$ into the PLM again. Accordingly, we use $h_{j/q_i}^s$ to denote the new representation of the $j$-th schema item when $q_i$ is masked out.

Formally, we measure the distance between $h_i^q$ and $h_{j/q_i}^s$ to induce the correlation between the schema item $s_j$ and the question token $q_i$ as follows:

$$f(q_i, s_j) = d(h_i^q, h_{j/q_i}^s)$$

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

$$d_{Euc}(q_i, s_j) = \|h_i^q - h_{j/q_i}^s\|_2$$

3.2 Poincaré Probe

Euclidean space has intrinsic difficulties in modeling complex data [4]. To better model the heterogeneous relational structures between the NL query and the database schema, we devise a Poincaré probe, which probes schema linking information from PLMs in the hyperbolic space that is expected to better capture linguistic hierarchies encoded in contextualized representations [40, 49]. As revealed in [6], the hyperbolic space enables vector comparison with much smaller distortion compared with the Euclidean space. In addition, recent work [6, 30, 40] demonstrates that the hyperbolic space may reflect some properties of graph naturally.

The Poincaré Ball. In this paper, we employ the standard Poincaré ball, which is a special model of hyperbolic spaces, to capture the difference between $h_{j/q_i}^s$ and $h_j^s$. Before introducing the Poincaré Probe, we first review basic concepts of the standard Poincaré ball following Ganea et al. [18]. Formally, for a point $x$ in the hyperbolic space, the standard Poincaré ball model is defined as $\mathbb{D}^n$, where $\mathbb{D}^n = \{x \in \mathbb{R}^n \mid \|x\|^2 < 1\}$ is a Riemannian manifold and $g_S^\infty = (\lambda_x)^2 I_n$ is the metric tensor. We formulate $\lambda_x = 2/(1 - \|x\|^2)$ as the conformal factor. Here, $n$ denotes the dimension size.

Feature Projection. To compare the feature vectors learned by PLMs in the hyperbolic space, we first use the exponential mapping function $g_S^\infty(\cdot)$ to project the embeddings into the hyperbolic space.

\footnote{Wu et al. [52] proposed a similar two-step perturbed masking. The difference is that they first masked out one input token $\{s_i\}$ and then masked out a token pair $\{s_i, s_j\}$. Then the correlation was obtained by comparing $h_{i/q_i}^s$ and $h_{j/q_i}^s$. This method heavily relies on the MLM prediction ability of the PLMs (e.g., BERT). We tried it in our preliminary experiments, but it performed poorly. We conjecture that our input data (i.e., questions and schemas) differs from the PLM’s pre-training data, and the PLM struggles to make reasonable schema predictions without further finetuning.}
We conduct extensive experiments on three benchmark datasets.

$\begin{align*}
\text{Spider} & \quad \text{is a large-scale cross-domain zero-shot text-to-SQL benchmark. We follow the common practice to report the exact match accuracy on the development set, as the test set is not publicly available.}
\end{align*}$

$\begin{align*}
\text{DK} & \quad \text{is a human-curated dataset based on Spider, a challenging variant of the Spider development set, with focus on evaluating the model understanding of domain knowledge.}
\end{align*}$

$\begin{align*}
\text{SYN} & \quad \text{is another challenging variant of Spider. SYN is constructed by manually modifying NL questions in Spider using synonym substitution, which aims to simulate the scenario where users do not know the exact schema words in the utterances.}
\end{align*}$
To have a better analysis on how Proton help capture relational knowledge from PLMs, we carefully conduct error checking in terms of schema linking on the Spider benchmark. We analyze the errors made by previous works [5, 51] and classify them into four categories: World Knowledge Error, Semantic Understanding Error, Type Error and Inference Error. We observe that Proton can successfully solve most of these bad cases which the previous methods fail to address. Due to the limited space, we only report one or two representative examples for each error category in Figure 4. From the results, we have the following observations.

First, we find that most of wrong predictions are due to the lack of world knowledge [31]. For example, as shown in the first example in Figure 4, the rule-based semantic linking with exact text matching can not predict "Brazil" as a country name, and it also fails to understand that "republic" is a government form in the second example. Theoretically, PLMs can be seen as an external knowledge resource by pre-training on large-scale corpora, and previous models with PLMs can identify that the word "republic" is related to "government form" by referring to PLMs. However, previous methods with PLMs still fail to capture these reference linking [5, 31, 51]. This is because only using the embeddings of PLMs cannot effectively elicit relational knowledge from PLMs. In contrast, our probing method can successfully elicit such world (relational) knowledge, and is portable for practical use.

Second, one kind of error categories is caused by failing to capture semantic relations between words and tables/columns. The rule-based schema linking often chooses columns/tables that exactly occur in the query via exact string matching. As shown in the fourth example in Figure 4, the rule-based schema linking method links the word "maker" to the column "car_makers.Maker" while the correct choice should be the column "car_names.Make". Similarly, the third example fails to identify that the word "highschooler" is a synonym of "students". This can be solved by considering in-depth semantic information of the query instead of only word occurrence. Our probing method can easily solve this kind of problem.

Third, some questions may contain more than one entity words that can match the table names. The rule-based schema linking could choose the wrong columns that match the entity names occurred in the question. For example, in the fifth example, the rule-based schema linking predicts the wrong column "Address" in the question. We find that Proton can largely improve the performance of RAT-SQL + Grappa, up to 7.9% on the DK benchmark and 13.5% on SYN benchmark. The reason may be that RAT-SQL + Grappa is not well designed for schema linking, while our Hyperbolic Proton can effectively capture the relational information between the NL query and the database schema by probing relational knowledge from PLMs.

### 5.2 Schema Linking Performance Analysis

To have a better analysis on how Proton help capture relational knowledge from PLMs, we carefully conduct error checking in terms of schema linking on the Spider benchmark. We analyze the errors made by previous works [5, 51] and classify them into four categories: World Knowledge Error, Semantic Understanding Error, Type Error and Inference Error. We observe that Proton can successfully solve most of these bad cases which the previous methods fail to address. Due to the limited space, we only report one or two representative examples for each error category in Figure 4. From the results, we have the following observations.

First, we find that most of wrong predictions are due to the lack of world knowledge [31]. For example, as shown in the first example in Figure 4, the rule-based semantic linking with exact text matching can not predict "Brazil" as a country name, and it also fails to understand that "republic" is a government form in the second example. Theoretically, PLMs can be seen as an external knowledge resource by pre-training on large-scale corpora, and previous models with PLMs can identify that the word "republic" is related to "government form" by referring to PLMs. However, previous methods with PLMs still fail to capture these reference linking [5, 31, 51]. This is because only using the embeddings of PLMs cannot effectively elicit relational knowledge from PLMs. In contrast, our probing method can successfully elicit such world (relational) knowledge, and is portable for practical use.

Second, one kind of error categories is caused by failing to capture semantic relations between words and tables/columns. The rule-based schema linking often chooses columns/tables that exactly occur in the query via exact string matching. As shown in the fourth example in Figure 4, the rule-based schema linking method links the word "maker" to the column "car_makers.Maker" while the correct choice should be the column "car_names.Make". Similarly, the third example fails to identify that the word "highschooler" is a synonym of "students". This can be solved by considering in-depth semantic information of the query instead of only word occurrence. Our probing method can easily solve this kind of problem.

Third, some questions may contain more than one entity words that can match the table names. The rule-based schema linking could choose the wrong columns that match the entity names occurred in the question. For example, in the fifth example, the rule-based schema linking predicts the wrong column "Address" in the question.
instead of the correct column "current_address_id". In addition, in some special cases, when the column name exactly matches the keyword in the question, the rule-based schema linking can not distinguish it clearly, as shown in the sixth example. This problem often occurs when the question contains keywords such as average, maximum, and minimum, etc.

Finally, the rule-based schema linking cannot solve the difficult cases that require strong inference ability. Taking the seventh example as an example, the word "accepted" in the question implicitly links the column "decision", which cannot be identified by exact string matching. Our PROTON model can solve these difficult cases by using semantic information learned from PLMs.

5.3 Case Study

We use four exemplary cases selected from DK and SYN benchmarks to demonstrate the effectiveness of our model qualitatively. Table 4 shows the SQL queries generated by the best baseline LGESQL and our PROTON model, where the first two cases are from the SYN benchmark and the last two cases are from the DK benchmark. From the results, we can observe that PROTON can generate correct SQL queries when dealing with challenging scenarios such as synonym substitution. For instance, in the first case, when replacing the schema-related word "type" with its synonym "category" in NL question, LGESQL fails to identify the correct column PetType.

Figure 5 shows the visualization of correlation matrices obtained by rule-based feature engineering and our probing technique. In the right two sub-figures, the rule-based technique (exact string matching) cannot capture the alignment between the word "category"
and the schema item PetType. While Proton can easily catch such semantic similarity with the help of elicited knowledge from PLMs, it thus generates the correct SQL query. Similarly, in the left two sub-figures, Proton successfully links the domain knowledge word “French” to column Citizenship, while LGESQL fails to identify the column Citizenship without explicit mentions. We believe that Proton can probe rich semantic and relational knowledge from large-scale PLMs, facilitating schema linking in text-to-SQL parsing.

6 RELATED WORK

Text-to-SQL Parsing. Recently, meaningful advances emerged on the encoder [3, 9, 25, 26], decoder [10, 27, 53] and table-based pre-training models [35, 42, 48, 54, 56] on Spider benchmark[57]. In particular, Lei et al. [31] pointed out that the schema linking module in the encoder was a crucial ingredient for successful prediction. To tackle the problem, Yu et al. [55] incorporated prior knowledge of column types and schema linking as additional input features. Guo et al. [20] used heuristic rules to construct intermediate representation. Rui et al. [46] used the co-attention mechanism to measure similarity between NL tokens with schema tokens. The recent method RAT-SQL [51] utilized a relational graph attention to handle various pre-defined relations and further considered both local and non-local edge features. To tackle the robustness problem in a more realistic setting, Gan et al. [16] proposed to use different data augmentation techniques including data annotation and adversarial training. Wang et al. [50] proposed a model-agnostic meta-learning based training objective to boost out-of-domain generalization of text-to-SQL models. Scholak et al. [47] propose PICARD, a method for constraining auto-regressive decoders of language models through incremental parsing. Different from these methods, we are the first to explore the potential knowledge stored in PLMs to help the model perform better schema linking and further improve the generalization of the model.

Probing PLMs. The success of PLMs has led to a large number of studies investigating and interpreting the rich knowledge that PLMs learn implicitly during pre-training [21, 22, 29, 45]. One typical approach is to probe PLMs with a small amount of learnable parameters considering a variety of linguistic properties, including morphology [2], word sense [44], syntax [12, 24], world knowledge [41] and semantics [34]. Another line of work is motivated to probe PLMs in an unsupervised and parameter-free fashion [32, 52]. Our work generally follows this line and exploits an unsupervised probing technique to extract relational knowledge for the downstream text-to-SQL parsing task. Liu et al. [36] proposed the ETA model to explore the grounding capabilities of PLMs. The proposed erasing-then-awakening trains a concept classification module by human-crafted supervision. Then, it erases tokens in a question to obtain the concept prediction confidence differences as pseudo alignment. Finally, it awakens latent grounding from PLMs by applying pseudo alignment as supervision. This method requires human-crafted label as supervision, which could not be easily obtained in most tasks. Furthermore, the additional trainable parameters may cause failures to adequately reflect differences in representations [23].

Different from previous methods, Proton does not need any extra labels or supervision, which is not limited to specific tasks. Our method follows an unsupervised technique, which makes sure all the relational knowledge is extracted from PLMs. In addition, Proton utilizes a direct graph to represent the relational knowledge and can better extract the relational information within the input.

7 CONCLUSION

In this paper, we proposed a probing technique to probe schema linking information between the NL query and the database schema from large-scale PLMs, which improved the generalization and robustness of the text-to-SQL parsing models. In addition, a Poincaré distance metric was devised to measure the difference between two vectors in the hyperbolic space, capturing the heterogeneous relational structures between the NL query and the database schema. Experimental results on three benchmark datasets demonstrated that our method substantially outperformed strong baselines and set state-of-the-art performance on three text-to-SQL benchmarks.

ACKNOWLEDGMENTS

This work was partially supported by National Natural Science Foundation of China (No. 61906185), Youth Innovation Promotion Association of CAS China (No. 2020357), Shenzhen Science and Technology Innovation Program (Grant No. KQTD20190929172835662), Shenzhen Basic Research Foundation (No. JCYJ20210324115614039 and No. JCYJ2020010913441941). This work was supported by Alibaba Group through Alibaba Innovative Research Program.
Table 5: Exact matching accuracy by varying the levels of difficulty of the inference data on the development sets of DK, SYN and Spider.

| Model                                | DK        | SYN       | Spider     |
|--------------------------------------|-----------|-----------|------------|
| RAT-SQL                              |           |           |            |
| RAT-SQL+Euclidean PROTON             | 69.0      | 68.9      | 87.9       |
| RAT-SQL+Hyperbolic PROTON            | 71.8      | 78.2      | 88.3       |
| LGESQL                               | 74.5      | 79.4      | 91.9       |
| LGESQL+Euclidean PROTON              | 72.8      | 81.5      | 91.9       |
| LGESQL+Hyperbolic PROTON             | 75.5      | 81.9      | 92.7       |

Table 6: Comparison of the inference time in seconds on 1034 SYN samples.

|                     | Model  | Model+Euclidean | Model+Hyperbolic |
|---------------------|--------|------------------|------------------|
| LGESQL              | 878(s) | 979(s)           | 975(s)           |
| RAT-SQL             | 1162(s)| 1255(s)          | 1387(s)          |

[57] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingming Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In EMNLP.

[58] Victor Zhong, Caiming Xiong, and R. Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. CoRR abs/1709.00103 (2017).

A RESULTS ON COMPLEX QUERIES

The three benchmarks provide four different difficulty levels of samples. We investigate the detailed model performance and have further insights on how PROTON can help complex queries. Table 5 shows the exact match accuracy by varying the levels of difficulty of the data. From the results, we can observe that PROTON can boost the performance of SOTA text-to-SQL parsers (RAT-SQL/LGESQL) across almost all different difficulty levels on the three benchmark datasets. It suggests that PROTON can lead to more significant accuracy improvements compared to RAT-SQL/LGESQL. For example, LGESQL with Hyperbolic PROTON, which better captures the relational knowledge, achieves the highest score on most cases. In particular, PROTON gains much better performance on the extremely hard samples than the baselines, verifying that the harder samples require better schema linking for correct text-to-SQL parsing.

B COMPUTATIONAL COST

We investigate the computational cost of baseline methods and our PROTON model in inference. All these models are run on a desktop machine with a single NVIDIA Tesla V100 GPU. In Table 6, we report the inference time on 1034 SYN samples with the batch size of 1. PROTON has a slightly slower inference speed than the base models (LGESQL and RAT-SQL). For example, on average, the inference time of PROTON with Poincaré probe increases by 0.09s on each sample compared with LGESQL, which is acceptable in practice.