SUN: Exploring Intrinsic Uncertainties in Text-to-SQL Parsers

Bowen Qin[1,2,✊], **Lihan Wang**[1,2,✊], **Binyuan Hui**[3,✊], **Bowen Li**[3], **Xiangpeng Wei**[3]
Binhua Li[3], **Fei Huang**[3], **Luo Si**[3], **Min Yang**[1†], **Yongbin Li**[3†]
[1] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
[2] University of Chinese Academy of Sciences
[3] DAMO Academy, Alibaba Group
{bw.qin, lh.wang1, min.yang}@siat.ac.cn
{binyuan.hby, xiangpeng.wxp, binhua.lbh, shuide.lyb}@alibaba-inc.com

Abstract

This paper aims to improve the performance of text-to-SQL parsing by exploring the intrinsic **un**certainties in the neural network based approaches (called **SUN**). From the data uncertainty perspective, it is indisputable that a single SQL can be learned from multiple semantically-equivalent questions. Different from previous methods that are limited to one-to-one mapping, we propose a data uncertainty constraint to explore the underlying complementary semantic information among multiple semantically-equivalent questions (many-to-one) and learn the robust feature representations with reduced spurious associations. In this way, we can reduce the sensitivity of the learned representations and improve the robustness of the parser. From the model uncertainty perspective, there is often structural information (dependence) among the weights of neural networks. To improve the generalizability and stability of neural text-to-SQL parsers, we propose a model uncertainty constraint to refine the query representations by enforcing the output representations of different perturbed encoding networks to be consistent with each other. Extensive experiments on five benchmark datasets demonstrate that our method significantly outperforms strong competitors and achieves new state-of-the-art results. For reproducibility, we release our code and data at https://github.com/AlibabaResearch/DAMO-ConvAI/tree/main/sunsql.

1 Introduction

Text-to-SQL parsing (Zettlemoyer and Collins, 2012; Liang et al., 2013; Zhong et al., 2017; Qin et al., 2022) aims at converting a natural language (NL) question to its corresponding structured query language (SQL) in the context of a relational database (Schema). 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 to a broader range of non-technical users. Therefore, text-to-SQL parsing has attracted increasing attention from both academic and industrial communities recently due to its broad applications in question answering, conversational search interaction, and so on.

Although significant efforts have been devoted to text-to-SQL parsing (Chen et al., 2021; Yu et al., 2018a; Wang et al., 2020a; Cao et al., 2021; Scholak et al., 2021) with advanced deep models and architectures by learning black-box mappings between input NL questions and output SQLs, there are still several technical challenges for accurate and robust text-to-SQL parsing. First, previous models are generally learned to fit the simplified one-to-one mapping relationship, where only one NL question is used as the input, and the appropriate rest are ignored. However, there exists data uncertainty (Ott et al., 2018; Wei et al., 2020) in text-to-SQL parsing, i.e., one output SQL may correspond to multiple semantically-equivalent NL questions. At inference time, the semantic parser trained on the one-to-one parallel data struggles to deal with adequate variations of the training queries. Second, there is often structural information (dependence) (Xiao and Wang, 2019; Zhang et al., 2022) among the weights of neural networks. One challenge in training neural semantic parsers is that such models may overfit the training data since these models only seek a point estimate for their weights, failing to quantify weight (model) uncertainty. For text-to-SQL parsing, the model uncertainty brings difficulty in obtaining the encoded representations that can best describe the input data distribution and provide an robust mapping between NL questions and SQL queries.

To alleviate the aforementioned challenges, in

✊ Equal contribution.
† Corresponding authors.

Proceedings of the 29th International Conference on Computational Linguistics, pages 5298–5308
October 12–17, 2022.

this paper, we propose a generic training approach SUN, which explores the data and model uncertainties in text-to-SQL parsing. First, we propose a data uncertainty constraint to explore the underlying complementary semantic information among multiple semantically-equivalent queries and learn comprehensive feature representations with strong expressive ability. In particular, we summarize multiple semantically-equivalent source questions into a closed semantic region which is then used to complement the model to generate better SQL queries with comprehensive semantics. Second, to improve the generalizability and stability of neural text-to-SQL parsers, we propose a model uncertainty constraint to refine the query representations by enforcing the output representations of different perturbed encoding networks to be consistent with each other. Concretely, we impose the consistency on multiple networks perturbed with dropout for the same input NL question.

We summarize our main contributions as follows. (1) We propose a data uncertainty constraint, which aims to explicitly capture comprehensive semantic information among multiple semantically-equivalent NL questions, and enhance the hidden representations with this complementary information for generating better SQL, thus improving the robustness of neural text-to-SQL parsers. (2) We employ a model uncertainty constraint to encourage high similarity between the output representations of two perturbed encoding networks for the same input NL question, improving the generalizability and stability of neural text-to-SQL parsers. (3) Experiments on five benchmark datasets demonstrate that the proposed SUN method outperforms the strong competitors by a substantial margin. ***It is noteworthy that our method is model-agnostic and potentially applicable for any text-to-SQL parsers with deep network architectures.***

2 Related Work

Text-to-SQL Parsing Text-to-SQL parsing, a subtask of semantic parsing, aims at converting a NL question to its corresponding SQL query in the context of a relational database (schema). Inspired by the success of deep learning, neural text-to-SQL models based on the sequence-to-sequence (Seq2Seq) framework have dominated the research field of text-to-SQL parsing (Guo et al., 2019; Wang et al., 2020b; Zhong et al., 2020; Hui et al., 2021a,b, 2022; Cao et al., 2021; Wang et al., 2022). The general idea behind these methods is to construct an encoder to encode the input question together with related table schema and leverage a decoder to generate the target SQL based on the output of the encoder. For example, IRNet (Guo et al., 2019) is a representative neural text-to-SQL parser, which leveraged two separate BiLSTMs with self-attention mechanism (Vaswani et al., 2017) to encode the NL question and table schema. Subsequently, the graph-based approaches have been proposed, which use relational graph attention networks to deal with the schema entities and question words with structured reasoning. For instance, RATSQL (Wang et al., 2020a), SMBOP (Rubin and Berant, 2021) and RaSaP (Huang et al., 2021) defined a question-schema graph and employed the relation-aware self-attention mechanism (Shaw et al., 2018) in the encoding process to jointly learn representations of question words, schema items and edge relations. LGESQL (Cao et al., 2021) further constructed an edge-centric graph to update the edge features and designed graph pruning to determine the golden schema items related to the NL question.

Recently, some methods have leveraged the powerful pre-training capabilities of T5 (Raffel et al., 2019) to generate SQL queries. Different from graph-based methods, T5-based approaches (Scholak et al., 2021) adopt the transformer-based architecture for both encoder and decoder and do not need pre-defined graphs, schema linking relations, and grammar-based decoder. PICARD (Scholak et al., 2021) is a representative T5-based method which constrained auto-regressive decoders of language models through incremental parsing. The impressive experimental results verify the ability of the T5-based methods for text-to-SQL parsing.

Uncertainty Modeling in NLP Uncertainty quantification is an important approach to building robust AI systems. In the field of natural language processing, there are several works (Kendall et al., 2015; Xiao and Wang, 2019; Zhang et al., 2019; Shen et al., 2019; Wei et al., 2020; Zhang et al., 2021; Hu and Khan, 2021) which investigate the effects of quantifying uncertainties in various NLP tasks. For example, Zhang et al. (2019) applied a dropout-entropy method to measure uncertainty learning for text classification. Xiao and Wang (2019) showed that explicitly modeling uncertainties via Monte-Carlo dropout (Gal and Ghahramani,

2016) could enhance model performances of several NLP tasks. Su et al. (2018) introduced a series of continuous latent variables to model underlying semantics of source sentences in neural machine translation. Wang et al. (2019) proposed to quantify the confidence of NMT model predictions based on model uncertainty to better cope with noise in synthetic corpora. Wei et al. (2020) considered the intrinsic uncertainty by representing multiple source sentences into a closed semantic region. To our best knowledge, our proposed method is the first attempt to explore uncertainty in text-to-SQL parsing.

3 Preliminaries

3.1 Problem Definition

Given a natural language question Q and the corresponding database schema $S = \langle T, C \rangle$, text-to-SQL parsing aims to generate a SQL query Y based on Q and S. More specifically, the question $Q = \{q_1, q_2, \cdots, q_{|Q|}\}$ is a sequence of tokens, and the schema S consists of tables $T = \{t_1, t_2, \cdots, t_{|T|}\}$ and columns $C = \{c_1, c_2, \cdots, c_{|C|}\}$. Each table t_i contains m_t words $(t_{i,1}, t_{i,2}, \cdots, t_{i,m_t})$ and each column name $c_j^{t_i}$ in table t_i contains m_c words $(c_{j,1}^{t_i}, c_{j,2}^{t_i}, \cdots, c_{j,m_c}^{t_i})$. We use $I = \langle Q, T, C \rangle$ to denote an input for the text-to-SQL parser.

3.2 Text-to-SQL parser

Currently, advanced text-to-SQL parsers are centred around two types of approaches: the graph-based approaches (Wang et al., 2020a; Huang et al., 2021; Cao et al., 2021) and the T5-based approaches (Scholak et al., 2021; Xie et al., 2022), both of which adopt the encoder-decoder framework for implementation.

Graph-based Methods Formally, the input question and database schema are constructed as a single direct graph in the pre-processing phase: $G = \langle V, E \rangle$, where $V = Q \cup T \cup C$ denotes the node set that contains three different node types (question, table, and column) and E is the edge set depicting *pre-existing* relations for question tokens and schema items. To obtain the initial representation for every node in the graph, the recent graph-based methods, *e.g.*, LGESQL (Cao et al., 2021), first flatten all question words and schema items into a sequence and feed the sequence $I = \langle Q, T, C \rangle$ into large-scale pre-trained language models (PLMs) to learn word vectors. The learned word vectors are then passed into a subword attentive pooling layer and three Bi-LSTMs according to the node types to get the node representations for the graph G.

After that, the graph-based approaches adopt an encoder, which consists of a stack of relational graph attention network (RGAT) (Wang et al., 2020b) layers, to learn complex interaction over schema items as well as question words and output the final contextual representation X_I for input I.

In the decoding process, the graph-based methods usually adopt the grammar-based syntactic neural decoder to generate the abstract syntax tree (AST) of the target query Y in the depth-first traversal order. The output at each decoding timestep is either (i) an `APPLYRULE` action that expands the current non-terminal node in the partially generated AST, or (ii) an `SELECTTABLE` or `SELECTCOLUMN` action that chooses certain schema item. The readers can refer to (Wang et al., 2020a) for more implementation details.

T5-based Methods The T5-based approaches leverage the powerful pre-training capabilities of T5 to generate SQL queries, which directly fine-tune the downstream corpora with the standard objective for text generation without any specifically designed modules. Specifically, they take the sequential concatenation of the question words and schema item names as input to the T5 encoder and generate the corresponding SQL using the T5 decoder. Both encoder and decoder are composed of multi-layer Transformer blocks.

To ensure that the output SQL is grammatically correct, T5-based approaches such as PICARD (Scholak et al., 2021) implement rule-based constrained decoding, achieving competitive performance with less invasiveness and better compatibility. PICARD is an incremental parsing method for constrained decoding and can be compatible with any existing auto-regressive language model decoder and vocabulary—including, but not limited to, those of large pre-trained transformers. Different from the graph-based approaches that generally restrict the auto-regressive decoding process to tokens that can correctly parse to abstract syntax trees during training, constrained decoding used in T5-based approaches operates directly on the output of the language model and applies the characteristic of target SQL to help reject inadmissible tokens at each decoding step at inference time.

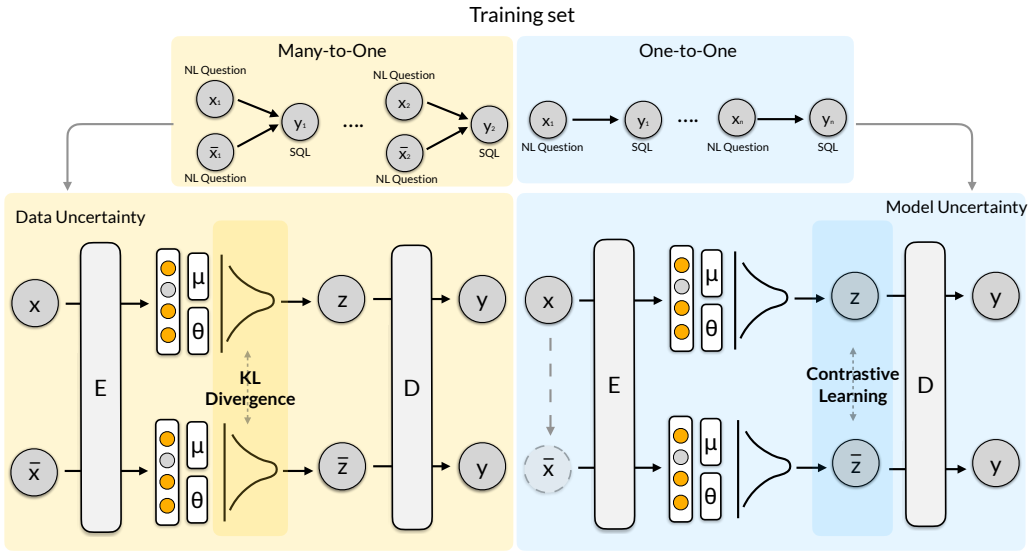

Figure 1: Overview of the proposed SUN method. There are many-to-one and one-to-one cases in the text-to-SQL training data. For many-to-one samples, they are modelled by data uncertainty, and for one-to-one samples, they are modelled by model uncertainty. The E means the encoder and D means the decoder of the graph-based or T5-based methods.

4 Proposed Method

Intrinsic Uncertainty Previous works have focused on the inductive bias through model design, while the intrinsic data uncertainty and model uncertainty are underexplored. As shown in Figure 1, **many-to-one** cases (multiple NL questions can correspond to one SQL query) exist in the text-to-SQL training data, which are caused by the inherent uncertainty of natural language (Ott et al., 2018; Wei et al., 2020). In such **many-to-one** scenario, each input sample I has a semantically-equivalent sample $\overline{I}$ in the training set, where these two input instances correspond to the same SQL query. In addition, there are also some instances in the training set that do not have semantically-equivalent samples (referred to **one-to-one** cases), where the model uncertainty becomes critical (Xiao and Wang, 2019; Zhang et al., 2019). In this work, we propose SUN, a generic training approach for text-to-SQL parsing, to mitigate the data uncertainty and model uncertainty simultaneously. SUN consists of two primary components: (i) a data uncertainty constraint that summarizes multiple semantically-equivalent NL questions into an abstract semantic region and (ii) a model uncertainty constraint that encourages high similarity between the output representations of two perturbed encoding networks for the same NL question.

Global Semantic Representation Learning Without the loss of generality, we use the encoder of the text-to-SQL parser to transform each input I into a contextual representation X_I. Then, we pass the contextual representation X_I to a convolutional layer with a pooling operation to obtain the final global semantic representation H_I of the input I. For Simplicity, we represent the whole process of learning global semantic representations as $H_I = \texttt{Encode}(I)$.

4.1 Data Uncertainty Constraint

For the many-to-one scenario, we expect to encourage high similarity between the representations of two semantically-equivalent NL questions. To this end, we propose a data uncertainty constraint to summarize the semantically-equivalent input I and $\overline{I}$ into a closed semantic region. We first obtain global semantic representations $H_I = \texttt{Encode}(I)$ and $H_{\overline{I}} = \texttt{Encode}(\overline{I})$ for two semantically-equivalent input samples I and $\overline{I}$. To learn compact and abstract representation without spurious association, we assume that there are two latent semantic variables $\mathbf{z}$ and $\overline{\mathbf{z}}$ associated with H_I and $H_{\overline{I}}$ respectively, which are sufficient for generating the target SQL query and eliminate the redundant details, so as to reduce the sensitivity of the learned representations. That is, both $\mathbf{z}$ and $\overline{\mathbf{z}}$ maintains all information which is shared by H_I and $H_{\overline{I}}$, inspired by the intuition that two semantically-

equivalent samples provide the same predictive information. Formally, we assume that $\mathcal{P}(\mathbf{z}|H_I)$ and $\mathcal{P}(\overline{\mathbf{z}}|H_{\overline{I}})$ are the distributions of the latent semantic variables $\mathbf{z}$ and $\overline{\mathbf{z}}$, which are modeled by Normal distributions parametrized with (μ,$\bar{\mu}$) and (σ, $\bar{\sigma}$). Formally, we define the two semantic distributions as follows:

$$\mathcal{P}(\mathbf{z} \mid H_I) \sim \mathcal{N}(\mu(H_I), \sigma^2(H_I)\mathbf{I}) \quad (1)$$

$$\mathcal{P}(\overline{\mathbf{z}} \mid H_{\overline{I}}) \sim \mathcal{N}\left(\bar{\mu}(H_{\overline{I}}), \bar{\sigma}^2(H_{\overline{I}})\mathbf{I}\right) \quad (2)$$

where $\mathbf{I}$ denotes the all-ones vector. μ (or $\bar{\mu}$) and σ (or $\bar{\sigma}$) are computed via fully-connected neural networks based on the global semantic representations H_I (or $H_{\overline{I}}$) as:

$$\mu(H_I) = H_I \cdot \mathbf{W}_\mu + \mathbf{b}_\mu \quad (3)$$

$$\log \sigma^2(H_I) = H_I \cdot \mathbf{W}_\sigma + \mathbf{b}_\sigma \quad (4)$$

$$\bar{\mu}(H_{\overline{I}}) = H_{\overline{I}} \cdot \mathbf{W}_\mu + \mathbf{b}_\mu \quad (5)$$

$$\log \bar{\sigma}^2(H_{\overline{I}}) = H_{\overline{I}} \cdot \mathbf{W}_\sigma + \mathbf{b}_\sigma \quad (6)$$

where $\mathbf{W}_\mu$, $\mathbf{W}_\sigma$ denote the projection parameters and $\mathbf{b}_\mu$, $\mathbf{b}_\sigma$ denote the bias terms.

Inspired by the reparameterization techniques used in (Kingma et al., 2014; Zhang et al., 2016), the sampled latent representations z and $\overline{z}$ for $\mathbf{z}$ and $\overline{\mathbf{z}}$ are then obtained as follows for efficient gradient computation :

$$z = \mu(H_I) + \sigma(H_I) \odot \epsilon \quad (7)$$

$$\overline{z} = \bar{\mu}(H_{\overline{I}}) + \bar{\sigma}(H_{\overline{I}}) \odot \epsilon \quad (8)$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ is a parameter to introduce noise and $\odot$ denotes an element-wise product.

The goal of the data uncertainty constraint is to summarize multiple source questions that share the same meaning into a closed semantic region. *In order to enforce the distribution $\mathcal{P}(\mathbf{z}|H_I)$ to be close to $\mathcal{P}(\overline{\mathbf{z}}|H_{\overline{I}})$, we formulate the data uncertainty constraint loss $\mathcal{L}_{\mathbf{DU}}$ by minimizing KL divergence between $\mathcal{P}(\mathbf{z}|H_I)$ and $\mathcal{P}(\overline{\mathbf{z}}|H_{\overline{I}})$ as*:

$$\mathcal{L}_{\mathbf{DU}} = \mathbf{KL}\left(\mathcal{P}(\overline{\mathbf{z}} \mid H_{\overline{I}}) || \mathcal{P}(\mathbf{z} \mid H_I)\right) \quad (9)$$

4.2 Model Uncertainty Constraint

To improve the robustness of text-to-SQL parsers, we consider the model uncertainty for the one-to-one questions which do not have semantically-equivalent questions in training set by taking advantage of the data augmentation technique. Specifically, the input data I goes through the forward pass of the encoding network twice with dropout to produce two-view representations of the input I. Since the dropout (Hinton et al., 2012) operator randomly drops units from the model, the two forward passes indeed produce two distinct semantic representations of input I. Inspired by the model uncertainty learning in text feature space (Zhang et al., 2019), we then use contrastive learning (Gao et al., 2021; Yan et al., 2021) to pull together the two-view representations of the same input question produced by dropout and push apart the semantic representations of different questions in the same batch. *Formally, we formulate the model uncertainty constraint loss $\mathcal{L}_{\text{MU}}$ as*:

$$\mathcal{L}_{\text{MU}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{s(z_i^1, z_i^2)}}{e^{s(z_i^1, z_i^2)} + \sum_{j=1, j\neq i}^{N} e^{s(z_i^1, z_j^1)}} \quad (10)$$

where s denotes a cosine similarity function $s(z_i^1, z_i^2) = z_i^1 \cdot z_i^2 / \|z_i^1\| \|z_i^2\|$, the superscript in z_i^1 and z_i^2 indicates the view index, N indicates the number of training samples in a mini-batch. The model uncertainty constraint encourages the consistency between different views of semantic representation from the same input while enforcing the discrepancy between unrelated question pairs. By reducing the gap between the sub-models that contain different weight correspondences due to the randomness of the dropout mechanism, the robustness of the representation of the text-to-SQL parsers can be further enhanced.

4.3 Uncertainty-aware Semantic Representation

We further augment the contextual representation X_I of the input sample I obtained by the encoder with the corresponding uncertainty-aware latent representation z through the learnable gate g as:

$$g = \text{sigmoid}\left(z \cdot \mathbf{W}_z + X_I \cdot \mathbf{W}_x\right) \quad (11)$$

where $\mathbf{W}_z$, $\mathbf{W}_x$ denote the projection parameters. Then, we formulate the overall semantic representation $\mathbf{U} \in \mathbb{R}^{|V^n| \times d}$ by combining the contextual representation X_I and the uncertainty-aware latent representation z with the corresponding gate g as:

$$\mathbf{U} = \text{LayerNorm}\left(g \cdot z + (1 - g) \cdot X_I\right) \quad (12)$$

where $\mathbf{U}$ is used as the input to the decoder (grammar-based decoder or T5 decoder) of the text-to-SQL parser to output the target SQL as $Y = \texttt{Decode}(\mathbf{U})$. We define the standard training objective of the encoder-decoder framework as $\mathcal{L}_{\text{T2S}}$.

Model	EM	EX
IRNet + BERT	61.9	-
RAT-SQL + BERT	69.7	-
RAT-SQL + Grappa	73.4	-
GAZP + BERT	59.1	53.5
BRIDGE + BERT	65.5	59.9
BRIDGE v2 + BERT	70.0	68.3
SMBOP + GRAPPA	69.5	71.1
RAT-SQL+GAP+NatSQL	68.7	73.3
LGESQL + ELECTRA	75.1	-
w/ SUN	**76.8**	-
PICARD + T5-Large	69.1	72.9
w/ SUN	71.6	**75.4**

Table 1: Exact match (EM) and execution (EX) accuracy (%) on SPIDER benchmark.

Model	Dev.		Test
	EM	EX	EX
ETA + BERT	50.10	68.30	54.10
ALIGN	43.70	62.10	50.10
w/ BERT	48.40	67.70	54.30
w/ RoBERTa	50.93	70.92	58.37
w/ SUN	**52.93**	**71.95**	**59.34**

Table 2: Exact match (EM) and execution (EX) accuracy (%) on SQUALL dataset.

Joint Training Finally, we combine the standard negative likelihood loss $\mathcal{L}_{\text{T2S}}$ for SQL generation and the two uncertainty constraint loss functions ($\mathcal{L}_{\text{DU}}$ and $\mathcal{L}_{\text{MU}}$) to form the joint loss function $\mathcal{L}_{\text{total}}$ as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{T2S}} + \mathcal{L}_{\text{DU}} + \mathcal{L}_{\text{MU}} \quad (13)$$

5 Experimental Setup

5.1 Datasets

We conduct extensive experiments on five benchmark datasets for text-to-SQL parsing. (1) **SPIDER** (Yu et al., 2018b) is a large-scale cross-domain zero-shot text-to-SQL benchmark. It originally contains 8659 training examples across 146 databases in total. We follow the common practice to report the exact match accuracy and execution accuracy. (2) **SYN** (Gan et al., 2021a) is a challenging variant of SPIDER, which consists of 1034 evaluation examples. SYN is constructed by manually modifying NL questions in SPIDER using synonym substitution. (3) **DK** (Gan et al., 2021b) is constructed by selecting 535 samples from SPIDER dev set, where 270 pairs are the original SPIDER samples while the rest 265 pairs are modified by incorporating the domain knowledge. (4) **REALISTIC** (Deng et al., 2020) is a more realistic and challenging evaluation setting with explicit mentions of column names being manually removed. (5) **SQUALL** (Shi et al., 2020) is constructed by generating SQL queries of the English-language questions in `WIKITABLEQUESTIONS` (Pasupat and Liang, 2015) and manually align questions with corresponding SQLs. It consists of 15,622 examples which are split into training (9,032), development (2,246) and test (4,344) sets. All datasets employed in this paper are in English.

Model	SYN	DK	REALISTIC
GNN	23.6	26.0	-
IRNet	28.4	33.1	-
RAT-SQL	33.6	35.8	-
RAT-SQL + BERT	48.2	40.9	58.1
RAT-SQL + Grappa	49.1	38.5	59.3
LGESQL + ELECTRA	64.6	48.4	69.2
w/ SUN	**66.9**	**52.7**	**70.9**

Table 3: Exact match accuracy (%) on SYN, DK and REALISTIC benchmark.

5.2 Baseline Methods

We compare SUN with several strong baseline methods, including IRNet (Guo et al., 2019), RAT-SQL (Wang et al., 2020a), GAZP (Zhong et al., 2020), BRIDGE (Lin et al., 2020), SMBOP (Rubin and Berant, 2021), LGESQL (Cao et al., 2021) and PICARD (Scholak et al., 2021). Since SUN is model-agnostic and potentially applicable for any neural text-to-SQL parsers, we adopt LGESQL and PICARD, which are the state-of-the-art graph-based and T5-based methods respectively, as our base models to verify the universality of SUN. In addition, for SQUALL, we adopt the previous SOTA model ALIGN (Shi et al., 2020) with RoBERTa (Devlin et al., 2018) and ETA (Liu et al., 2021) as our base model.

5.3 Implementation Details

For the LGESQL, following (Cao et al., 2021), the hidden size of the graph attention network is set to 512 and the number of layers is set to 8. The number of heads in multi-head attention is 8 and the dropout rate is set to 0.2 for both the encoder and decoder. In the decoder, the dimension of hidden

Model	SPIDER					SYN				
	easy	medium	hard	extra	all	easy	medium	hard	extra	all
LGESQL+ELECTRA	91.9	78.3	64.9	**52.4**	75.1	79.4	67.9	**62.1**	36.1	64.6
w/ SUN	**92.3**	**80.3**	**70.7**	50.6	**76.8**	**79.8**	**72.3**	59.9	**41.4**	**66.9**
Model	DK					REALISTIC				
	easy	medium	hard	extra	all	easy	medium	hard	extra	all
LGESQL+ELECTRA	74.5	46.7	41.9	29.5	48.4	86.2	**77.8**	60.6	41.2	69.2
w/ SUN	**75.5**	**53.7**	**47.3**	**30.5**	**52.7**	**89.9**	77.3	**61.6**	**45.4**	**70.9**

Table 4: Exact matching accuracy by varying the levels of difficulty of the inference data on the of SPIDER, SYN, DK and REALISTIC.

Model	**SPIDER**	**SYN**	**DK**	**REALISTIC**
LGESQL+SUN	76.8	66.9	52.7	70.9
w/o $\mathcal{L}_{\text{DU}}$	75.8	65.8	51.2	69.5
w/o $\mathcal{L}_{\text{MU}}$	76.2	66.4	52.1	70.1

Table 5: Ablation results in terms of exact match accuracy on SPIDER, SYN, DK and REALISTIC.

state, action embedding and node type embedding are set to 512, 128 and 128, respectively. We use AdamW optimizer (Loshchilov and Hutter, 2017) with linear warmup scheduler and the warmup ratio of total training steps is 0.1. The learning rate is 1e-4 and the weight decay rate is 0.1. The batch size is set to be 20 and the maximum gradient norm is 5. The number of training epochs is 200. For the PICARD (Scholak et al., 2021), we follow the official implementation to fine-tune T5-large for 400 epochs. We use Adafactor optimizer (Shazeer and Stern, 2018) with a learning rate of 1e-4. For the ALIGN model, we employ SUN on the RoBERTa setting. The representation size is 1024. The decoder is implemented with 2-layer LSTM and the hidden size is set to 128. We adopt the Adam optimizer (Kingma and Ba, 2014) with learning rate of 0.001 and the dropout rate is set to be 0.3. Specifically, we employ kernels with window sizes ranging from 3 to 5 to obtain the global semantic representations used in the uncertainty measurement process in SUN.

6 Experimental Results

6.1 Main Results

Results on SPIDER Table 1 shows the exact match accuracy (EM) and execution accuracy (EX) scores of our method and compared baselines on the SPIDER dataset. We observe that SUN can bring substantial improvements, which achieves a notable gain of 1.7% on the exact match accuracy scores over the strongest baseline LGESQL. In addition, PICARD+SUN obtains significantly better results than the compared baseline methods on SPIDER dataset in terms of exact match accuracy and execution accuracy, which both achieve superior results of 2.5% improvement.

Results on more challenging and realistic settings To evaluate the effectiveness of SUN in more challenging and realistic settings, Table 3 illustrates the experimental results on SYN, DK and REALISTIC datasets. We can observe that SUN consistently and substantially surpasses the compared models by a noticeable margin on three datasets in terms of exact match accuracy. In particular, SUN achieves considerable improvement over LGESQL on all three datasets. For SYN, LGESQL with SUN outperforms LGESQL by 2.3% EM score, demonstrating that SUN can improve the robustness of text-to-SQL parsers to synonym substitution. An improvement of 4.3% EM score is observed on DK benchmark, verifying that SUN contributes to the generalization ability of text-to-SQL parsers to unseen domains. Furthermore, under the challenging REALISTIC setting where all explicit mentions of column names are removed, LGESQL+SUN also achieves a strong performance (70.9%) which is 1.7% higher than LGESQL.

Results on SQUALL Table 2 shows the experimental results on SQUALL. Since SQUALL does

Case 1. (SPIDER)		
Gold	SELECT T1.template_type_code, count(*) FROM Templates AS T1 JOIN Documents AS T2 ON T1.template_id = T2.template_id GROUP BY T1.template_type_code	✓
Question	*Show all template type codes and the number of documents using each type.*	
LGESQL	SELECT T1.template_type_code, count(*) FROM Templates AS T1 JOIN Documents AS T2 ON T1.template_id = T2.template_id GROUP BY T1.template_type_code	✓
w/ SUN	SELECT T1.template_type_code, count(*) FROM Templates AS T1 JOIN Documents AS T2 ON T1.template_id = T2.template_id GROUP BY T1.template_type_code	✓
SE_Question	*What are the different template type codes, and how many documents use each type?*	
LGESQL	SELECT Templates.Template_Type_Code , COUNT(*) FROM Templates GROUP BY Templates.Template_Type_Code	✗
w/ SUN	SELECT T1.template_type_code, count(*) FROM Templates AS T1 JOIN Documents AS T2 ON T1.template_id = T2.template_id GROUP BY T1.template_type_code	✓
Case 2. (DK)		
Gold	SELECT count(*) FROM student AS T1 JOIN has_pet AS T2 ON T1.stuid = T2.stuid JOIN pets AS T3 ON T2.petid = T3.petid WHERE T1.sex = 'F' AND T3.pettype = 'dog'	
Question	*How many puppy pets are raised by female students?*	
LGESQL	SELECT count(*) FROM student AS T1 JOIN has_pet AS T2 ON T1.stuid = T2.stuid JOIN pets AS T3 ON T2.petid = T3.petid WHERE T1.sex = 'F' AND T3.pettype = 'dog'	✓
w/ SUN	SELECT count(*) FROM student AS T1 JOIN has_pet AS T2 ON T1.stuid = T2.stuid JOIN pets AS T3 ON T2.petid = T3.petid WHERE T1.sex = 'F' AND T3.pettype = 'dog'	✓
SE_Question	*Find the number of puppy pets that are raised by female students (with sex F).*	
LGESQL	SELECT COUNT(*) FROM Pets JOIN Has_Pet JOIN Student WHERE Student.Sex = 'F'	✗
w/ SUN	SELECT count(*) FROM student AS T1 JOIN has_pet AS T2 ON T1.stuid = T2.stuid JOIN pets AS T3 ON T2.petid = T3.petid WHERE T1.sex = 'F' AND T3.pettype = 'dog'	✓

Table 6: Case study: cases are sampled from SPIDER, and DK. SE indicates semantically-equivalent.

not contain target SQL queries in the test set, we merely present the execution accuracy score on test set. Overall, SUN significantly improves the performance on both the dev and test sets of SQUALL, achieving the gains of 2.0% EM score and 1.03% EX score on dev set, and 1.97% EX score on test set over the strongest baseline ALIGN+RoBERTa on SQUALL.

6.2 Results on Complex Queries

We investigate the performance of SUN on different queries. The SQL queries in the SPIDER, SYN, DK and REALISTIC benchmarks are divided into four levels (i.e., easy, medium, hard, extra hard) based on their difficulty, where the difficulty is defined based on the number of SQL components. Table 4 summarizes the results of the four benchmarks with four levels of difficulty. From the results, we can observe that SUN can boost the performance of LGESQL across almost all different difficulty levels on the four datasets. In particular, LGESQL+SUN obtains much better performance on the extremely hard samples than LGESQL by a large margin. For example, LGESQL with SUN shows 4.7%, 1.0% and 4.2% improvements on the extra hard samples in SYN, DK and REALISTIC respectively.

6.3 Ablation Study

To analyze the impact of two kinds of uncertainties in SUN, we also conduct an ablation test by discarding the data uncertainty constraint (denoted as w/o $\mathcal{L}_{\mathrm{DU}}$) and the model uncertainty constraint (denoted as w/o $\mathcal{L}_{\mathrm{MU}}$), respectively. The ablation results are summarized in Table 5. As expected, both uncertainty constraints contribute great improvements to SUN. For example, the performance of SUN w/o $\mathcal{L}_{\mathrm{DU}}$ decreases by 1.1/1.0/1.5 points on the SYN/SPIDER/DK dev sets, verifying that the data uncertainty constraint is essential for improving the model performance. In addition, the performance of SUN decreases by 0.8/0.5 points on REALISTIC/SYN evaluation sets when removing the model uncertainty constraint.

6.4 Case Study

In this section, we present two cases sampled from SPIDER and DK to demonstrate the effectiveness of SUN qualitatively. As illustrated in Table 6, we report the original questions, the semantically-equivalent variant of the original questions (SE-question), the gold SQL queries, and the SQL queries generated by LGESQL and LGESQL+SUN. From the results, we can observe that LGESQL+SUN can generate more accurate SQL queries than LGESQL. Taking the first case as an example, although both LGESQL and LGESQL+SUN can generate accurate SQL queries given the question, LGESQL fails to correctly understand its semantically-equivalent question and thus generates inappropriate SQL where the necessary mention of database schema "*Document*" is ignored. In contrast, LGESQL+SUN generates the same correct SQL for both the original question

and its semantically-equivalent question by improving the robustness of the text-to-SQL parsers with data and model uncertainty constraints. We observe similar trends on the DK dataset. Concretely, LGESQL+SUN can generate consistent SQL for two semantically-equivalent questions that have very different expressions, while LGESQL cannot handle such cases.

The effectiveness of SUN according to the observed results illustrated in Table 6 demonstrates that the exploration of underlying complementary semantic information in SUN helps text-to-SQL parsers understand the different variants of expressions and generate better SQL queries with comprehensive semantics.

7 Conclusion

In this paper, we proposed a novel SUN method to explore intrinsic uncertainties in text-to-SQL parsing. First, we devised a data uncertainty constraint to capture complementary semantic information among multiple semantically-equivalent questions and thus improve the robustness of the text-to-SQL parsers. Second, a model uncertainty constraint was leveraged to refine the representations by encouraging high similarity between the output representations of two perturbed encoding networks for the same input question. Experimental results on five benchmark datasets showed that SUN significantly outperformed the compared methods.

8 Acknowledges

My acknowledges: 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. JCYJ20200109113441941).

References

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. Lgesql: Line graph enhanced text-to-sql model with mixed local and non-local relations. In *ACL*.

Zhi Chen, Lu Chen, Yanbin Zhao, Ruisheng Cao, Zihan Xu, Su Zhu, and Kai Yu. 2021. Shadowgnn: Graph projection neural network for text-to-sql parser. In *Proc. of NAACL*.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2020. Structure-grounded pretraining for text-to-sql. *arXiv preprint arXiv:2010.12773*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proc. of ICML*.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John Robert Woodward, Jinxia Xie, and Pengsheng Huang. 2021a. Towards robustness of text-to-sql models against synonym substitution. In *ACL*.

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021b. Exploring underexplored limitations of cross-domain text-to-sql generalization. In *EMNLP*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. In *ACL*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Yibo Hu and Latifur Khan. 2021. Uncertainty-aware reliable text classification. In *Proc. of KDD*.

Junyang Huang, Yongbo Wang, Yongliang Wang, Yang Dong, and Yanghua Xiao. 2021. Relation aware semi-autoregressive semantic parsing for nl2sql. *arXiv preprint arXiv:2108.00804*.

Binyuan Hui, Ruiying Geng, Qiyu Ren, Binhua Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, Pengfei Zhu, and Xiaodan Zhu. 2021a. Dynamic hybrid relation exploration network for cross-domain context-dependent semantic parsing. In *AAAI*.

Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Bowen Li, Jian Sun, and Yongbin Li. 2022. S^2SQL: Injecting syntax to question-schema interaction graph encoder for text-to-sql parsers. In *ACL*.

Binyuan Hui, Xiang Shi, Ruiying Geng, Binhua Li, Yongbin Li, Jian Sun, and Xiaodan Zhu. 2021b. Improving text-to-sql with schema dependency learning. *arXiv preprint arXiv:2103.04399*.

Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. 2015. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Proc. of NeurIPS*.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. *arXiv preprint arXiv:2012.12627*.

Qian Liu, Dejian Yang, Jiahui Zhang, Jiaqi Guo, Bin Zhou, and Jian-Guang Lou. 2021. Awakening latent grounding from pretrained language models for semantic parsing. In *ACL*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. In *ICML*, pages 3956–3965. PMLR.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, et al. 2022. A survey on text-to-sql parsing: Concepts, methods, and future directions. *arXiv preprint arXiv:2208.13629*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Ohad Rubin and Jonathan Berant. 2021. Smbop: Semi-autoregressive bottom-up semantic parsing. In *Proc. of NAACL*.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. In *EMNLP*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proc. of ICML*.

Aili Shen, Daniel Beck, Bahar Salehi, Jianzhong Qi, and Timothy Baldwin. 2019. Modelling uncertainty in collaborative document quality assessment. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*.

Tianze Shi, Chen Zhao, Jordan Boyd-Graber, Hal Daumé III, and Lillian Lee. 2020. On the potential of lexico-logical alignments for semantic parsing to sql queries. *arXiv preprint arXiv:2010.11246*.

Jinsong Su, Shan Wu, Deyi Xiong, Yaojie Lu, Xianpei Han, and Biao Zhang. 2018. Variational recurrent neural machine translation. In *Proc. of AAAI*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Proc. of NeurIPS*.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020a. RAT-SQL: relation-aware schema encoding and linking for text-to-sql parsers. In *ACL*.

Kai Wang, Weizhou Shen, Yunyi Yang, Xiaojun Quan, and Rui Wang. 2020b. Relational graph attention network for aspect-based sentiment analysis. *arXiv preprint arXiv:2004.12362*.

Lihan Wang, Bowen Qin, Binyuan Hui, Bowen Li, Min Yang, Bailin Wang, Binhua Li, Fei Huang, Luo Si, and Yongbin Li. 2022. Proton: Probing schema linking information from pre-trained language models for text-to-sql parsing. In *KDD*.

Shuo Wang, Yang Liu, Chao Wang, Huanbo Luan, and Maosong Sun. 2019. Improving back-translation with uncertainty-based confidence estimation. *arXiv preprint arXiv:1909.00157*.

Xiangpeng Wei, Heng Yu, Yue Hu, Rongxiang Weng, Luxi Xing, and Weihua Luo. 2020. Uncertainty-aware semantic augmentation for neural machine translation. *arXiv preprint arXiv:2010.04411*.

Yijun Xiao and William Yang Wang. 2019. Quantifying uncertainties in natural language processing tasks. In *Proc. of AAAI*.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *arXiv preprint arXiv:2201.05966*.

Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. Consert: A contrastive framework for self-supervised sentence representation transfer. *arXiv preprint arXiv:2105.11741*.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir R. Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proc. of NAACL*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018b. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *EMNLP*.

Luke S Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*.

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *EMNLP*.

Chen Zhang, Lei Ren, Fang Ma, Jingang Wang, Wei Wu, and Dawei Song. 2022. Structural bias for aspect sentiment triplet extraction. In *COLING*.

Linhai Zhang, Chao Lin, Deyu Zhou, Yulan He, and Meng Zhang. 2021. A bayesian end-to-end model with estimated uncertainties for simple question answering over knowledge bases. *Computer Speech & Language*.

Xuchao Zhang, Fanglan Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2019. Mitigating uncertainty in document classification. *arXiv preprint arXiv:1907.07590*.

Victor Zhong, Mike Lewis, Sida I Wang, and Luke Zettlemoyer. 2020. Grounded adaptation for zero-shot executable semantic parsing. *arXiv preprint arXiv:2009.07396*.

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