DeLVe-SQL: Decoupled Latent Variable Models for Few-Shot Text-to-SQL

Anonymous ACL submission

Abstract

Few-shot single-table text-to-sql tasks present considerable challenges due to the constraints of limited training data. Existing approaches primarily transform this problem into column-based classification tasks and utilize self-training methods to leverage unlabeled texts with pseudo-labels. The critical challenge, however, lies in selecting high-quality pseudo-labels and incorporating them effectively into model training. Past self-training techniques selected pseudo SQL predictions based on the probabilities yielded by column-specific classifiers. This approach may not align well with the original queries, especially given the limited performance of the few-shot classifier. To address these limitations, we introduce a novel approach DeLVe-SQL: a latent variable model specifically designed for few-shot text-to-sql tasks. This model effectively decouples textual and SQL semantics via distinct latent variables, enhancing the classifier's performance. Moreover, we apply an additional GPT2 decoder to take into account the reconstruction probabilities of the original query given pseudo SQL predictions, providing a more refined weighting of pseudo-labels. Our experiments, conducted on both open-domain and domain-specific benchmarks, demonstrate that our proposed method delivers promising results, outperforming existing methods in few-shot scenarios.

1 Introduction

Text-to-sql generation is a critical component in the field of natural language processing, enabling users to interact with databases using natural language inputs (Zhong et al., 2017; Yu et al., 2018, 2019; Wang et al., 2020a; Yu et al., 2021; Xu et al., 2022; Li et al., 2023c). In this landscape, both single-table and multi-table querying approaches serve distinct purposes and have broad applications. The single-table text-to-sql generation approach focuses on queries within a specific table, offering a simplified way to extract information. This is often suitable for scenarios where data is neatly structured within individual tables (Sun et al., 2020; Chen et al., 2021; Guo et al., 2022) where the intricacy of multiple table interactions is not necessary.

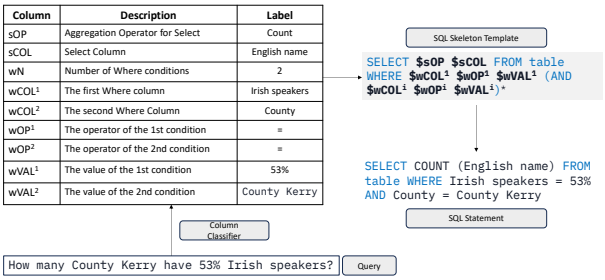

Figure 1: An example of single table text-to-sql.

Figure 1 illustrates this concept with an example where the SQL statement is uniformly fitted into a skeleton template with six column attributes, such as the column for SELECT (Xu et al., 2017; Hwang et al., 2019). This formulation translates the SQL statement generation into classification problems, streamlining the learning process in fully supervised settings. It is especially effective when utilizing strong tabular pretrained encoders (Yin et al., 2020; Herzig et al., 2020; Yu et al., 2021; Deng et al., 2022; Giaquinto et al., 2023), as demonstrated in the open-domain WikiSQL dataset (Zhong et al., 2017).

Despite fully supervised training, few-shot single-table text-to-sql (Chang et al., 2019; Wang et al., 2021; Chen et al., 2021; Guo et al., 2022) is crucial in real-world scenarios where new tables or databases have limited labeled data. It enables models to adapt and generate accurate SQL queries with minimal training examples, reducing the need for extensive manual annotation. By leveraging prior knowledge and handling few-shot scenarios, text-to-SQL models become more scalable, versatile, and adaptable to different domains without requiring extensive retraining.

There are limited efforts to solve this task.

Chang et al. (2019) pioneered the study of zero-shot text-to-sql generalization by proposing an auxiliary mapping task between language query and table columns, which still requires heavy supervision. Chen et al. (2021) and Wang et al. (2021) apply coarse-grained meta-learning to adapt models for unseen tables with table content information understanding. However, their improvements are limited due to their column-agnostic model parameter updating strategy. Guo et al. (2022) propose a meta self-training approach (MST-SQL) with fine-grained meta learning dependent on table columns for few-shot single-table text2sql, achieving significant improvement. Sun et al. (2023) fine-tunes a large private PALM[1] language model for few-shot text-to-sql task, which is expensive.

In this paper, we investigate the MST-SQL method proposed by Guo et al. (2022) for few-shot text-to-SQL tasks. Among the existing few-shot approaches, MST-SQL stands out for its fine-grained meta-learning dependent on table columns, enabling it to overcome limitations observed in previous work. This unique approach not only aligns with the complexities of few-shot scenarios but also offers a promising avenue for significant improvement. While self-training techniques have the potential to utilize unlabeled texts with pseudo-labels (Liu et al., 2022), they encounter two challenges in the few-shot scenario. First, ensuring high-quality pseudo SQL predictions via column-based classifiers become difficult in few-shot settings, which may lead to diverged synthetic predictions and affect the effectiveness of self-training. Secondly, the alignment between natural language query and the table columns are more challenging due to noisy pseudo labels, leading to optimization and generalization issues.

Our work aims to address these issues by introducing a decoupled latent variable model, dubbed DeLVe-SQL. DeLVe-SQL decouples **textual** and SQL column **label** semantics via distinct latent variables, which has not been previously utilized in the field of text-to-SQL. The textual semantic refers to the meaning conveyed at the level of the natural language text. It encompasses the way different phrases or expressions in a query can signify the same underlying intent. For instance, phrases like "how many", "count", or "the total number of" all share a common semantic purpose for the "count" aggregator in SQL, even though they are linguistically distinct. The label semantic pertains to the structural and syntactic knowledge encapsulated in the SQL statement. It deals with how certain keywords or phrases in the natural language query directly relate to specific SQL components (like aggregators, conditions, column names, etc.). For example, the phrase "last year" can be mapped to the "max" aggregator in certain situations. The label latent variables are only responsible for column classification. The textual latent variables are sent to a frozen GPT2 (Radford et al., 2019) decoder for reconstructing the language query. The advantages of DeLVe-SQL are manifold. The latent variables implicitly augment the training data via sampled latent variables and stabilize the training process via regularization terms between prior and posterior distributions. By decoupling the semantics and employing separate latent variables, our model ensures that the learning of each aspect doesn't interfere with each other, leading to better performance. Furthermore, the effective weighting of pseudo-labels, guided by the reconstruction probabilities, reduces the noise introduced by incorrect pseudo-labels, improving the overall quality of generated SQLs.

To validate our approach, we conduct comprehensive experiments on two general single-table text-to-sql benchmarks including both WikiSQL (Zhong et al., 2017) and ESQL (Chen et al., 2021), which are commonly used in the few-shot settings. The results demonstrate that DeLVe-SQL achieves promising results, surpassing existing methods in few-shot scenarios. Our codes will be released.

2 Related Work

Text-to-SQL Text-to-SQL has been explored meticulously across both single-table and multi-table contexts, employing datasets like WikiSQL (Zhong et al., 2017) and Spider (Yu et al., 2018). The multi-table scenarios, bringing forth their unique set of complexities and challenges, have seen substantial advancements (Scholak et al., 2021; Wang et al., 2020b; Qi et al., 2022; Li et al., 2023a,b) within fully supervised setups.

For single-table text-to-SQL, Seq2SQL (Zhong et al., 2017) interprets this task through the lens of sequence generation, utilizing Seq2Seq neural networks (Sutskever et al., 2014). A coarse-to-fine decoding method that initially predicts a linearized sketch, subsequently decoding the full SQL based on the sketch, is introduced by Dong and Lapata

[1] https://ai.google/discover/palm2/

(2018). SQLNet (Xu et al., 2017) introduces an innovative approach that partitions the text-to-SQL task into six discrete sub-tasks, each tasked with predicting a component in the SQL query. Further research has explored enhanced encoders (Hwang et al., 2019), schema linking strategies (He et al., 2019; Ma et al., 2020; Hui et al., 2021; Xu et al., 2022), and enhanced decoding methods (Lyu et al., 2020; Lin et al., 2020), and tabular pre-trained models (Herzig et al., 2020; Yin et al., 2020; Yu et al., 2021; Deng et al., 2021; Giaquinto et al., 2023).

Few-shot Learning for Text-to-SQL. Pioneering the study of zero-shot text-to-sql, Chang et al. (2019) propose an auxiliary mapping task to explicitly model relationships between natural language entities and table column names, serving as a supportive model and regularization term, which enhances the model's generalization capacity, as evidenced by a noteworthy improvement in generalizability on a zero-shot subset test. Chen et al. (2021) employ a meta-learning strategy, leveraging table content information to manage zero-shot tables without necessitating additional manual annotations. Yang et al. (2022) introduce sequential prompting, which decomposes the text-to-sql task into sub-level problems at the sub-clause level, facilitating few-shot compositional semantic parsing. Meanwhile, Guo et al. (2022) use meta self-training, specifically crafted for single-table text-to-SQL tasks, and uses self-training to navigate the complexities of few-shot text-to-SQL problems. This approach also incorporates a column specificity meta-learning algorithm to comprehend universal concepts. Our work takes a leaf from Guo et al. (2022) but adds a twist by using decoupled latent variables to provide advantages to both the few-shot column classifier and the self-trainer.

The recent surge in instruction-tuned large language models has unveiled potent capabilities across zero-shot and few-shot text-to-sql scenarios (Liu et al., 2023; Liu and Tan, 2023; Tai et al., 2023; Pourreza and Rafiei, 2023; Chang and Fosler-Lussier, 2023), showcasing formidable results on benchmarks like Spider (Yu et al., 2018) and BIRD (Li et al., 2023c). Sun et al. (2023) also explores finetuning a private large-scale language model (PaLM) for text-to-SQL tasks. SQLCoder-70B[2] even shows the state-of-the-art text-to-SQL performances in SQL-eval after fine-tuning on close-sourced data. Although comparisons with such models are not considered in our work due to potential unknown data contamination issues and high computing costs, the decoupled latent variables may serve a vital role in selecting effective demonstrations for in-context learning with models like ChatGPT and GPT4 (Pourreza and Rafiei, 2023), providing avenues for future research.

Decoupled Latent Variable Models. Yin et al. (2018) introduce the StructVAE model, implementing tree-structured latent variable models for semi-supervised semantic parsing. A bi-level latent model for few-shot compositional text generation is proposed by Xia et al. (2020), where domain and action are overseen by two latent variables. Li et al. (2022) integrate disentangled priors in variational auto-encoders for low-resource task-specific natural language generation, incorporating conditional priors for latent content and label spaces. Meanwhile, Singh and Jamali-Rad (2022) showcase a variational inference network that isolates class-specific attributes from image context and semantics in few-shot image classification. Although our decoupled latent variable models draw from similar principles to these works, they are specifically adapted for the text-to-SQL task, which involves dealing with multiple classifiers and hierarchical structures. Moreover, our approach combines the latent variable models with GPT2 (Radford et al., 2019), providing a distinct methodology compared to prior research.

3 Decoupled Latent Variable Models for Few-shot Text-to-SQL

We first discuss the few-shot text-to-SQL setting in Section 3.1, then describe the training objectives with latent variable modeling in Section 3.2.

3.1 Few-shot Text-to-SQL

Our approach incorporates self-training for few-shot text-to-SQL by following the settings of Guo et al. (2022). The training data is divided into labeled data, denoted as $\mathcal{L}$, and unlabeled data, denoted as $\mathcal{U}$. Labeled data $\mathcal{L}$ consists of a set of tuples $\mathcal{L}_i = (q_i, T_i, y_i)$, where q_i represents the i-th natural language query, T_i denotes the table corresponds to q_i, and y_i represents the correct SQL label for the corresponding q_i over T_i. T_i consists of column-header pairs $\{(h_{i,j}, \mathcal{C}_{i,j})\}_{j=1}^{M_i}$, where M_i is the number of columns of T_i, $h_{i,j}$ is the j-th column header of T_i and $\mathcal{C}_{i,j}$ is the cell values under $h_{i,j}$. Unlabeled data U comprises a set of

[2] https://github.com/defog-ai/sqlcoder

tuples $\mathcal{U}_i = (q_i, T_i)$ without explicit SQL labels. y_i follows a standardized format as shown in Figure 1, where tokens marked with $ represent slots to be filled, and * indicates zero or more occurrences.

Following the previous studies (Hwang et al., 2019; Lyu et al., 2020), generating each y_i can be divided into six tasks: 1) decide the column (`$sCOL`, **SC**) to be included in the `SELECT` clause; 2) decide the aggregation function (`$sOP`, **SA**) within the `SELECT` clause; 3) decide the number (`$wN`, **WN**) of conditions in the `WHERE` clause; 4) decide the column (`$wCOL`, **WC**) for the i-th condition in the `WHERE` clause; 5) decide the operator (`$wOP`, **WO**) for the i-th condition in the `WHERE` clause; 6) decide the value (`$wVAL`, **WV**) for the i-th condition in the `WHERE` clause.

3.2 Decoupled Latent Variable Model

The overall training objective is to maximize the log-likelihood on both the label data $\mathcal{L}$ and the unlabeled data $\mathcal{U}$, which is given by

$$\mathcal{O} = \sum_{i}^{|\mathcal{L}|} \log p(y_i, q_i, T_i) + \sum_{j}^{|\mathcal{U}|} \log p(q_j, T_j) \quad (1)$$

We introduce two latent variables $\mathbf{z}_t$ and $\mathbf{z}_l$ for decoupling the textual and label semantics into respective terms in Eq 2. Here for simplicity, we use the first term $p(y_i, q_i, T_i)$ as an example to show the whole process, which is formally given by

$$p(y_i, q_i, T_i) = \int\int p(y_i, q_i, T_i|\mathbf{z}_t, \mathbf{z}_l) p(\mathbf{z}_t, \mathbf{z}_l) d\mathbf{z}_l d\mathbf{z}_t \quad (2)$$

Direct learning objective is intractable in general. Therefore we turn to the evidence-lower-bound (ELBO) using a variational posterior distribution $\phi(\mathbf{z}_t, \mathbf{z}_l \mid q_i, T_i)$ to approximate $p(y_i, q_i, T_i)$,

$$\begin{aligned} p(y_i, q_i, T_i) &= \iint p(y_i, q_i, T_i|\, \mathbf{z}_t, \mathbf{z}_l)\, p(\mathbf{z}_t, \mathbf{z}_l)\, d\mathbf{z}_t\, d\mathbf{z}_l, \\ &= \mathbb{E}_{\phi(\mathbf{z}_t, \mathbf{z}_l \mid q_i, T_i)} \left[\frac{p(q_i, T_i \mid \mathbf{z}_t, \mathbf{z}_l) p(y_i \mid \mathbf{z}_l) p(\mathbf{z}_l) p(\mathbf{z}_t)}{\phi(\mathbf{z}_t, \mathbf{z}_l \mid q_i, T_i)} \right]. \\ &\approx \mathbb{E}_{\phi(\mathbf{z}_t, \mathbf{z}_l \mid q_i, T_i)} \left[\frac{p(q_i \mid \mathbf{z}_t, \mathbf{z}_l) p(y_i \mid \mathbf{z}_l) p(\mathbf{z}_l) p(\mathbf{z}_t)}{\phi(\mathbf{z}_t, \mathbf{z}_l \mid q_i, T_i)} \right]. \end{aligned}$$

Here $p(q_i, T_i \mid \mathbf{z}_t, \mathbf{z}_l)$ is the reconstruction probability to generate q_i and T_i based on $\mathbf{z}_t$ and $\mathbf{z}_l$. We simplify it by only reconstructing the query using $p(q_i \mid \mathbf{z}_t, \mathbf{z}_l)$. $p(y_i, q_i, T_i)$ is further given by,

$$\begin{aligned} &\log p(y_i, q_i, T_i) \\ &\geqslant \mathbb{E}_{\phi(\mathbf{z}_t, \mathbf{z}_l | q_i, T_i)} \left[\log \left(\frac{p(q_i \mid \mathbf{z}_t, \mathbf{z}_l) p(y_i \mid \mathbf{z}_l) p(\mathbf{z}_l) p(\mathbf{z}_t)}{\phi(\mathbf{z}_t, \mathbf{z}_l \mid q_i, T_i)} \right) \right], \\ &= \mathbb{E}_{\phi_2} \left[\mathbb{E}_{\phi_1} \left[\log \left(\frac{p(q_i \mid \mathbf{z}_t, \mathbf{z}_l) p(y_i \mid \mathbf{z}_l) p(\mathbf{z}_l) p(\mathbf{z}_t)}{\phi_2(\mathbf{z}_t \mid q_i, T_i) \phi_1(\mathbf{z}_l \mid q_i, T_i, \mathbf{z}_t)} \right) \right] \right]. \end{aligned}$$

Here $\phi(\mathbf{z}_t, \mathbf{z}_l \mid q_i, T_i)$ is decomposed into two cascade variational posterior distributions, namely $\phi_1(\mathbf{z}_l \mid q_i, T_i, \mathbf{z}_t)$ and $\phi_2(\mathbf{z}_t \mid q_i, T_i)$ for $\mathbf{z}_l$ and $\mathbf{z}_t$, respectively. And the ELBO objective is defined as

$$\begin{aligned} \mathcal{O}_{\text{ELBO}} = &- \mathbb{E}_{\phi_2} \mathbb{E}_{\phi_1} \left[\log p(q_i \mid \mathbf{z}_t, \mathbf{z}_l) + \log p(y_i \mid \mathbf{z}_l) \right] + \\ &D_{KL}\big(\phi_1(\mathbf{z}_l \mid q_i, T_i, \mathbf{z}_t) \| \, p(\mathbf{z}_l)\big) + \\ &D_{KL}\big(\phi_2(\mathbf{z}_t \mid q_i, T_i) \| \, p(\mathbf{z}_t)\big), \end{aligned} \quad (3)$$

where $p(\mathbf{z}_l)$ and $p(\mathbf{z}_t)$ are the two prior distributions for $\mathbf{z}_l$ and $\mathbf{z}_t$, respectively. Both the prior distributions and the variational posterior distributions are assumed to be Gaussian distribution, which can allow using reparmeterization trick for sampling latent variables and leading to closed KL divergence solutions for the last two terms defined in Eq 3.

Similarly, the log-likelihood of the unlabeled data $\log p(q_j, T_j)$ can be approximated by excluding the labeling term $\log p(y_i \mid \mathbf{z}_l)$ from Eq 3. Eq 3 shows the modeling process by treating y_i as a whole label. In practice, for each column label, we introduce two latent variables: one for the textual and the other for SQL column labels as shown in Figure 2. For example, the textual and label latent variables for the **SC** classifier are $\mathbf{z}_{sc,t}$ and $\mathbf{z}_{st,l}$, respectively.

4 Model Architecture

Figure 2 outlines the structure of our proposed method, consisting of four main submodules: 1) The Encoder focuses on learning the input representation; 2) The Latent Variable Sampler plays a pivotal role in drawing decoupled latent variables; 3) The Submodule Classifier is for column classification; 4) Finally, the Query Reconstruction module is responsible for reconstructing the queries.

4.1 Encoder

We utilize a RoBERTa Encoder (Liu et al., 2019) and take a similar approach to Hydranet (Lyu et al., 2020; Guo et al., 2022) by adopting column-wise input formation for the RoBERTa encoder. Specifically, the input pair (q, T) is broken down to m column-inputs, each represented as (q, h^i) for $i = 1, ..., m$. The RoBERTa model then generates hidden states for each (q, h^i) pair. This input sequence incorporates special tokens like the `[CLS]` symbol, column type t^i, column header h^i, table content, and the language query q by following Chen et al. (2021). The table content comprises n cells from $\mathcal{C}^i$ that hold the highest literal scores.

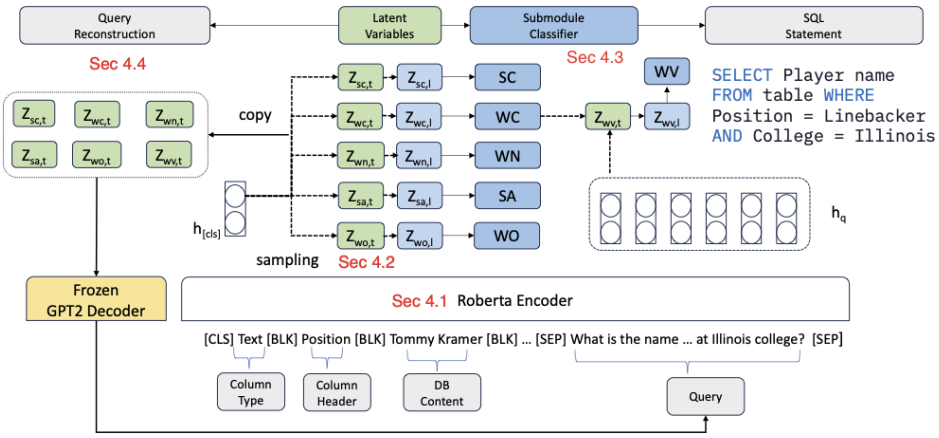

Figure 2: An overview of DeLVe-SQL. The query is "What is the name of the linebacker at Illinois college?".

$\mathbf{h}_q^k \in \mathbb{R}^d$ stands for the vector of the k-th token in q, and $\mathbf{h}_{[CLS]}^i \in \mathbb{R}^d$ which serves as the semantic vector of the entire column-input (q, h^i).

4.2 Latent Variable Sampler

Based on $\mathbf{h}_{[CLS]}^i$, we generate the decoupled latent variables. $\mathbf{h}_{[CLS]}^i$ is first encoded by a MLP,

$$\mathbf{r}_{[CLS]}^i = \text{MLP}(\mathbf{h}_{[CLS]}^i, \theta_{lv})$$

where $\theta_{lv} \in \{\theta_{\text{SC}}, \theta_{\text{WC}}, \theta_{\text{WN}}, \theta_{\text{SA}}, \theta_{\text{WO}}, \theta_{\text{WV}}\}$ and denotes the specific parameters for different columns. $\mathbf{r}_{[CLS]}^i$ is used to generate the mean $\boldsymbol{\mu}$ and the variance parameters $\boldsymbol{\sigma}$ of Gaussian distributions. Taking the **SC** task as an example, they are given by

$$\begin{aligned}\boldsymbol{\mu}_{i,sc,t} &= \mathbf{W}_{sc,t,\mu}\mathbf{r}_{[CLS],sc}^i + \mathbf{b}_{sc,t,\mu},\\ \boldsymbol{\sigma}_{i,sc,t}^2 &= \exp(\mathbf{W}_{sc,t,\sigma}\mathbf{r}_{[CLS],sc}^i + \mathbf{b}_{sc,t,\sigma})\end{aligned} \tag{4}$$

where $\mathbf{W}_{sc,\mu}$, $\mathbf{b}_{sc,\mu}$, $\mathbf{W}_{sc,\sigma}$, and $\mathbf{b}_{sc,\sigma}$ are model parameters. The textual latent variable is given by $\mathbf{z}_{i,sc,t} = \boldsymbol{\mu}_{i,sc,t} + \boldsymbol{\sigma}_{i,sc,t}^2 \cdot \boldsymbol{\epsilon}$. $\boldsymbol{\epsilon}$ is a random vector from a normal distribution. Similarly, we can generate the label latent variable by conditioning on both $\mathbf{r}_{[CLS],sc}^i$ and $\mathbf{z}_{i,sc,t}$ using concatenation $\oplus$, which is formally given by

$$\begin{aligned}\boldsymbol{\mu}_{i,sc,l} &= \mathbf{W}_{sc,l,\mu}[\mathbf{r}_{[CLS],sc}^i \oplus \mathbf{z}_{i,sc,t}] + \mathbf{b}_{sc,l,\mu},\\ \boldsymbol{\sigma}_{i,sc,l}^2 &= \exp(\mathbf{W}_{sc,l,\sigma}[\mathbf{r}_{[CLS],sc}^i \oplus \mathbf{z}_{i,sc,t}] + \mathbf{b}_{sc,l,\sigma})\\ \mathbf{z}_{i,sc,l} &= \boldsymbol{\mu}_{i,sc,l} + \boldsymbol{\sigma}_{i,sc,l}^2 \cdot \boldsymbol{\epsilon}\end{aligned} \tag{5}$$

4.3 Submodule Classifier

The submodule classifier uses the label latent variables for each column-input to tackle various subtasks. We train a separate MLP for each column classifier. We start with the essential SC task, ranking the columns based on relevance using

$$p_{\text{SC}}(h^i = \texttt{\$sCOL}|q) = \sigma(\mathbf{w}_{sc} \cdot \mathbf{z}_{i,sc,l}) \tag{6}$$

Here, σ represents the sigmoid function and $\mathbf{w}_{sc}$ is a trainable parameter. The column h^i with the highest p_{SC} is chosen as $\texttt{\$sCOL}$. Similarly, we calculate $P_{\text{WC}}(h^i \in \{\texttt{\$wCOL}^j\}|q)$ for WC using parameter $\mathbf{w}_{\text{WC}} \in \mathbb{R}^d$, and the top-$\mathcal{N}$ columns h^i with the highest P_{WC} are returned as $\texttt{\$wCOL}^j$. The number of WHERE conditions, $\mathcal{N}$, is predicted by

$$\begin{aligned}p_{\text{WN}}(\mathcal{N}^j|q, h^i) &= \text{softmax}(\mathbf{W}_{\text{WN}}[j,:] \cdot \mathbf{z}_{i,wn,l})\\ \mathcal{N} &= \text{argmax}_j(\sum_i \omega_i \cdot p_{\text{WN}}(\mathcal{N}^j|q, h^i))\end{aligned} \tag{7}$$

Here, $\mathbf{W}_{\text{WN}} \in \mathbb{R}^{n\times d}$ is a model parameter, and ω_i is the weight calculated by $\text{softmax}(\mathbf{w}_\omega \cdot \mathbf{z}_{i,wn,l})$. The remaining sub-tasks can be solved with $\texttt{\$sCOL}$ and $\texttt{\$wCOL}^j$. Specifically, $\texttt{\$wOP}^j$ is the operator $o^k \in O$ with the highest conditional probability, computed by

$$P_{\text{WO}}(o^k|q, \texttt{\$wCOL}^j) = \text{softmax}(\mathbf{W}_{\text{WO}}[k,:] \cdot \mathbf{z}_{i,wo,l,j})$$

where $\mathbf{W}_{\text{WO}} \in \mathbb{R}^{|O|\times d}$. Subsequently, for WV, x_q^k is selected as the start of $\texttt{\$wVAL}^j$ which possesses the highest probability

$$P_{\text{WV}}^{st}(x_q^k = st|q, \texttt{\$wCOL}^j) = \text{softmax}(\mathbf{h}_q^k \cdot (\mathbf{W}_{\text{WV}}^{st}\mathbf{z}_{i,wv,l}))$$

where $\mathbf{W}_{\text{WV}}^{st} \in \mathbb{R}^{d\times d}$, and $\mathbf{h}_q^k \in \mathbb{R}^d$ is the hidden vector of x_q^k. The end index of $\texttt{\$wVAL}^j$ is found in a similar way. Finally, we obtain the SQL program by filling all slots of the skeleton.

4.4 Query Reconstruction

This module creates natural language queries utilizing a set of sampled latent variables, denoted as $\mathbf{Z} = \{\mathbf{z}_{sc,t}, \mathbf{z}_{wc,t}, \mathbf{z}_{wn,t}, \mathbf{z}_{sa,t}, \mathbf{z}_{wo,t}, \mathbf{z}_{wn,t}\}$. These latent variables serve as a prefix for a static GPT2 decoder and are strategically masked to inhibit attention towards each other. The language modeling

loss, guided by the constructed query, is propagated back to these latent variables, thereby enabling their training to align correspondingly with the table columns.

In the context of self-training, the probability of query construction, given as $p_{\text{LM}}(q|\mathbf{Z})$, is employed to compute the confidence ψ of selecting pseudo-predictions. This is formulated as:

$$\psi = \sqrt[\mathcal{N}+2]{p_{\text{LM}}(q|\mathbf{Z}) p_{\text{SC}}(\texttt{\$sCOL}|q) \prod_{\texttt{\$wCOL}^j} p_{\text{WC}}(\texttt{\$wCOL}^j|q)}, \tag{8}$$

where the confidence ψ thoughtfully considers both text-to-sql and sql-to-text reconstruction probabilities. Specifically, it accounts for the classifier outputs $p_{\text{SC}}(\texttt{\$sCOL}|q)$ and $p_{\text{WC}}(\texttt{\$wCOL}^j|q)$, in relation to text-to-sql, and the sql-to-text reconstruction probability $p_{\text{LM}}(q|\mathbf{Z})$, furnishing a more robust estimation for pseudo-labels.

This approach integrates the merits of employing latent variables in guiding the decoder, as well as the strategic utility of self-training in generating high-confidence pseudo-labels. The use of static GPT-2 ventures to inject a more linguistically nuanced, contextually aware mechanism into the SQL query generation process, thereby nudging the generated queries to be more in alignment with the intended semantic meaning embedded within the natural language input. By juxtaposing text-to-sql and sql-to-text reconstruction probabilities, this module strives to curate an environment conducive for generating SQL queries that are not only syntactically correct but are also semantically coherent with respect to the input natural language queries.

5 Experiments

5.1 Data and Settings

Data We follow the experimental settings of Chen et al. (2021) and Guo et al. (2022) to evaluate our proposed models on both WikiSQL (Zhong et al., 2017) and ESQL (Chen et al., 2021). The data statistics is shown in Table 5 in Appendix A.1.

Evaluation We use the standard evaluation metrics, including both Logical Form (LF) and Execution (EX) accuracies. LF is for exact matching scores between the predicted SQL and the gold answer. EX is for comparing the executed results using the predicted SQLs with the results obtained by gold SQLs. See Appendix A.10 for more explanations.

Training settings The training settings are shown in Appendix A.4. The introduction of self-training is shown in Appendix A.2.

Method	WikiSQL (#shots)				ESQL (#shots)			
	1	2	3	4	5	10	15	20
SQLOVA	23.3	40.8	48.7	55.3	22.3	39.6	52.0	54.7
HydraNet	64.2	69.9	72.9	74.3	43.6	58.1	70.5	76.7
MC-SQL	52.0	62.9	71.0	73.8	36.5	53.2	60.5	67.4
BRIDGE	53.6	68.9	73.1	77.3	-	-	-	-
TABERT	57.5	67.4	71.2	72.5	-	-	-	-
GRAPPA	72.8	76.8	78.0	78.1	-	-	-	-
MST-SQL	78.4	80.5	82.1	83.2	55.3	67.4	76.7	80.5
DeLVe-SQL	**79.3**	**81.8**	**83.7**	**84.9**	**56.8**	**68.9**	**78.7**	**82.0**

Table 1: Few-shot LF accuracies. The baseline results are referred from MST-SQL (Guo et al., 2022) and the empty cells are due to that the corresponding baselines do not use ESQL.

5.2 Few-shot Results

Table 1 showcases the results in a few-shot learning scenario for both WikiSQL and ESQL databases. From this table, we can see that the performance of all methods improves as the number of shots increases. This is expected as more examples provide more information for the models to learn from. In WikiSQL, the DeLVe-SQL model consistently outperforms the other methods across different shot counts, with performance improvement as the shot count increases. For example, in the 1-shot setting, DeLVe-SQL achieves a score of 79.3, which is notably higher than the second best-performing model, MST-SQL, at 78.4. This trend continues in the 2-shot, 3-shot, and 4-shot scenarios, and DeLVe-SQL reaches a score of 84.9 in the 4-shot setting. The ESQL results follow a similar pattern, and DeLVe-SQL shows superior performance across different shot counts. For instance, in the 5-shot setting, DeLVe-SQL attains a score of 56.8, considerably outperforming MST-SQL's score of 55.3. As the number of shots increases to 10, 15, and 20, DeLVe-SQL's performance continues to surpass the other models.

5.3 Fully-supervised Results

The fully-supervised results are shown in Appendix A.5. Our proposed model, DeLVe-SQL, achieves a LF score of 87.0% and an EX score of 92.5% in the Dev set, and a LF score of 86.5% and an EX score of 92.1% in the test set, outperforming the other methods including MST-SQL.

6 Analysis

6.1 Ablation Study

We perform the ablation studies on WikiSQL test set stemming from its large-scale and diverse nature. With the complexity and assortment it offers,

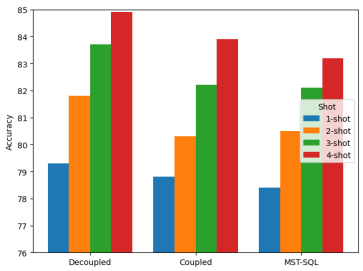

Figure 3: Decoupled v.s. Coupled latent variables on WikiSQL test set. The starting value at the y-axis is 76.

$\mathbf{z}_{sc}$	$\mathbf{z}_{wc}$	$\mathbf{z}_{wn}$	$\mathbf{z}_{sa}$	$\mathbf{z}_{wo}$	$\mathbf{z}_{wv}$	1-shot	2-shot	3-shot	4-shot
✓	✓	✓	✓	✓	✓	**79.3**	**81.8**	**83.7**	**84.9**
	✓	✓	✓	✓	✓	78.9	81.1	83.3	84.2
		✓	✓	✓	✓	78.5	80.9	82.8	84.1
			✓	✓	✓	78.7	81.2	83.1	83.9
		✓	✓		✓	78.5	80.8	83.2	83.9
		✓	✓	✓		78.3	80.7	82.4	83.4
✓	✓					78.9	81.4	83.0	84.3
✓	✓				✓	79.0	81.5	83.7	84.6

Table 2: The effect of different latent variables on WikiSQL test set.

WikiSQL provides a rigorously challenging and robust testing environment.

Decoupled v.s. Coupled Latent Variables In this study, we compare the decoupled v.s. coupled latent variables using few-shot learning on WikiSQL. For coupled latent variables, we set the texutal and label latent variables to be the same. Figure 3 shows the results. In all settings, the model with decoupled latent variables outperforms the one with coupled latent variables. For instance, in the 1-shot scenario, the model with decoupled latent variables achieves a score of 79.3, while the coupled one scores slightly lower at 78.8. This trend continues across the remaining shot scenarios, culminating in a score of 84.9 for the decoupled model in the 4-shot setting, versus 83.9 for the coupled model. Interestingly, the model with coupled latent variables still outperforms MST-SQL across all the three settings (except for 2-shot), which suggests that even coupled latent variable can provide advantages for few-shot text-to-sql.

Effect of Different Latent Variables In order to examine whether all of the latent variables for the six sub tasks are necessary, we design a probing method by gradually removing them from both the submodule classifier and the query deconstruction module on WikiSQL. Table 2 shows the effect of different latent variable combinations.

The results indicate that each of these variables can contribute separately to different aspects of SQL query generation. The performance decrease of $\mathbf{z}_{wn}, \mathbf{z}_{sa}, \mathbf{z}_{wo}$ is marginal, indicating that these variables are not critically dependent on any single one when contributing to the overall performance. Interestingly, the combination of $\mathbf{z}_{sc}$ and $\mathbf{z}_{wc}$ with $\mathbf{z}_{wv}$ achieves a similar result to the full model in the 4-shot scenario, suggesting that these three variables might be particularly important for the model performance. Removing all the three variables results in the largest decrease in performance. Our observation is that SC and WC are the most critical for SQL generation, which is consistent with the findings of Lyu et al. (2020) and Yin et al. (2020). In addition, we find that $\mathbf{z}_{wv}$ is quite important since it is directly related to the query content. In conclusion, while all six latent variables contribute to the model high performance, the model is quite robust to the removal of individual variables, and maintains a close performance to its best result with different combinations of the variables. Its robustness also demonstrates the distributed nature of the model learning and its ability to effectively make use of various subset information.

Confidence Estimation We investigate the role of three factors defined in Eq 8 in estimating the confidence of pseudo-labels. Table 3 shows the results. Table 3 indicates that the performance declines whenever any single factor is excluded from the process. In particular, omitting the query reconstruction term p_{LM} results in more obvious performance decrease. The 3-shot and 4-shot performances are decreased from 83.7% and 84.9% to 82.6% and 84.1%, respectively. The SC and WC classification probabilities, represented by p_{SC} and p_{WC} respectively, exhibit comparable performances when they are excluded.

Interestingly, when only one factor is in play, p_{LM} appears to be the most robust metric, and surpasses the baseline MST-SQL. However, exclusive reliance on p_{SC} or p_{WC} leads to severely compromised performance. These observations collectively suggest that a combination of all three factors yields the best performance, with p_{LM} proving to be the most crucial one among them.

6.2 Case Study

Table 4 delineates a comparative analysis of the output cases from MST-SQL, DeLVe-SQL, and the gold references, across four distinct queries. In the first case, both MST-SQL and DeLVe-SQL mirror the gold reference in their outputs, given that the columns and the aggregation queries are explicated lucidly within the question, facilitating

p_{LM}	p_{SC}	p_{WC}	1-shot	2-shot	3-shot	4-shot
✓	✓	✓	**79.3**	**81.8**	**83.7**	**84.9**
✓	✓		79.0	81.1	83.5	84.2
✓		✓	79.0	81.2	83.4	84.5
	✓	✓	78.7	81.0	82.6	84.1
✓			78.6	80.9	82.6	83.8
	✓		78.2	80.0	81.9	83.0
		✓	78.0	80.3	81.2	82.8

Table 3: The effect of confidence estimation on WikiSQL test set.

a straightforward translation into SQL.

Moving to the second case, MST-SQL misinterprets the column, opting for "Year" to sift through games for the 3DS platform. In contrast, DeLVe-SQL and the gold reference astutely select the "Platform(s)" column. This deviation possibly springs from MST-SQL's difficulty in discerning column semantics from its attributes, particularly when "Platform" isn't overtly stated in the originating query. DeLVe-SQL, enriched semantically with a GPT2 decoder, displays a heightened ability to comprehend implicit column references from the input query, likely attributing to its accurate column selection.

In the third case, MST-SQL manages to predict merely 2 out of 3 requisite columns, while DeLVe-SQL adeptly identifies all. This differential could potentially stem from MST-SQL's inability to semantically correlate 'wins total' with the 'Total_wins' column, suggesting a limitation in its semantic mapping capabilities.

For the final case, it's observable that MST-SQL omits the aggregation function MAX. Contrarily, DeLVe-SQL, even in the absence of terms like highest or largest, astutely outputs the correct aggregator function. This precision can potentially be attributed to DeLVe-SQL's capacity to identify that the term last semantically corresponds to a MAX aggregation operation over the 'Year' column, due to its latent semantic analysis.

In conclusion, these cases empirically show that DeLVe-SQL, with its adept context comprehension, skill in understanding aggregate statements, and proficiency in grasping multi-conditional scenarios, showcases a formidable and enhanced performance in text-to-SQL translation.

More analysis Appendix A.6 shows the ablation study of the query reconstruction module, which demonstrates that incorporating the query reconstruction module has a positive effect on the accuracy of both MST-SQL and DeLVe-SQL models. Appendix A.7 and Appendix A.8 show the significance test and model variances, respectively. Appendix A.9 shows the quality of generated queries.

Question: What is the smallest number of European Parliament sets when the international affiliation is global greens, egp?

Gold: `SELECT MIN European_Parliament_seats FROM table WHERE International_Affiliation = 'Global Greens, EGP';`

MST-SQL: `SELECT MIN European_Parliament_seats FROM table WHERE International_Affiliation = 'Global Greens, EGP'`

DeLVe-SQL: `SELECT MIN European_Parliament_seats FROM table WHERE International_Affiliation = 'Global Greens, EGP'`

Question: What 3ds game did Naohiko Aoyama direct?

Gold: `SELECT Title FROM table WHERE Platform(s) = 3ds AND Director = naohiko aoyama;`

MST-SQL: `SELECT Title FROM table WHERE Year = 3ds AND Director = naohiko aoyama;`

DeLVe-SQL: `SELECT Title FROM table WHERE Platform(s) = 3ds AND Director = naohiko aoyama;`

Question: Who was placed fourth when third was Beijing Guoan and the winner was Dalian Wanda and wins total was 4?

Gold: `SELECT fourth-placed FROM table WHERE Third-place = 'Beijing Guoan' AND Winners = 'Dalian Wanda' AND Total_wins = 4`

MST-SQL: `SELECT fourth-placed FROM table WHERE Third-place = 'Beijing Guoan' AND Winners = 'Dalian Wanda'`

DeLVe-SQL: `SELECT fourth-placed FROM table WHERE Third-place = 'Beijing Guoan' AND Winners = 'Dalian Wanda' AND Total_wins = 4`

Question: What was the last year that they had less than 16 points in class 500cc?

Gold: `SELECT MAX Year FROM table WHERE Points < 16 AND Class = 500cc`

MST-SQL: `SELECT Year FROM table WHERE Points < 16 AND Class = 500cc`

DeLVe-SQL: `SELECT MAX Year FROM table WHERE Points < 16 AND Class = 500cc`

Table 4: We compared DeLVe-SQL's outputs with the gold references and the outputs of MST-SQL.

7 Conclusion

This study addresses the challenges of few-shot single-table Text-to-SQL generation by introducing a decoupled latent variable model (DeLVe-SQL), effectively separating textual and SQL column label semantics. DeLVe-SQL not only enhances the model performance by ensuring independent learning for each part, but also reduces noise in pseudo-labels and improves SQL query quality. Experimental results on WikiSQL and ESQL benchmarks demonstrate that DeLVe-SQL outperforms existing methods in few-shot training scenarios. We perform extensive ablation studies to investigate the effective factors for few-shot text-to-sql learning, demonstrating that both the decoupled latent variable model and the improved confidence estimation contribute to the performance improvements. We hope that our method will encourage further research into robust and versatile Text-to-SQL models, particularly with limited labeled data.

8 Limitation

Our approach is specifically designed for single-table text-to-SQL tasks, where the queries involve a single table and do not consider more complex scenarios involving multiple tables or joins. The applicability of our approach is restricted to this specific domain and may not generalize well to more complex SQL query generation tasks.

Furthermore, in our approach, we solely utilize GPT2 as the decoder for query reconstruction. We do not incorporate other decoder models or architectures for this purpose. While GPT2 has shown promising performance in various natural language processing tasks, it is important to note that our approach does not explore the potential benefits or drawbacks of alternative decoder models in the context of query reconstruction.

References

Shuaichen Chang and Eric Fosler-Lussier. 2023. How to prompt llms for text-to-sql: A study in zero-shot, single-domain, and cross-domain settings.

Shuaichen Chang, Pengfei Liu, Yun Tang, Jing Huang, Xiaodong He, and Bowen Zhou. 2019. Zero-shot text-to-sql learning with auxiliary task.

Yongrui Chen, Xinnan Guo, Chaojie Wang, Jian Qiu, Guilin Qi, Meng Wang, and Huiying Li. 2021. Leveraging table content for zero-shot text-to-sql with meta-learning.

Naihao Deng, Yulong Chen, and Yue Zhang. 2022. Recent advances in text-to-SQL: A survey of what we have and what we expect. In *COLING*, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2021. Structure-grounded pretraining for text-to-SQL. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1337–1350, Online. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.

Robert Giaquinto, Zhang Dejiao, Benjamin Kleiner, Yang Li, Ming Tan, Parminder Bhatia, and Xiaofei Nallapati, Ramesh Ma. 2023. Multitask pretraining with structured knowledge for text-to-SQL generation.

Xinnan Guo, Yongrui Chen, Guilin Qi, Tianxing Wu, and Hao Xu. 2022. Improving few-shot text-to-sql with meta self-training via column specificity. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 4150–4156. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. 2019. X-sql: reinforce schema representation with context.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

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

Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *AAAI*.

Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. 2023b. Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing.

Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023c. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls.

Zhuang Li, Lizhen Qu, Qiongkai Xu, Tongtong Wu, Tianyang Zhan, and Gholamreza Haffari. 2022. Variational autoencoder with disentanglement priors for low-resource task-specific natural language generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10335–10356, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S. Yu. 2023. A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability.

Qi Liu, Zihuiwen Ye, Tao Yu, Linfeng Song, and Phil Blunsom. 2022. Augmenting multi-turn text-to-SQL

datasets with self-play. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5608–5620, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xiping Liu and Zhao Tan. 2023. Divide and prompt: Chain of thought prompting for text-to-sql.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. 2020. Hybrid ranking network for text-to-sql.

Jianqiang Ma, Zeyu Yan, Shuai Pang, Yang Zhang, and Jianping Shen. 2020. Mention extraction and linking for SQL query generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6936–6942, Online. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction.

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. Rasat: Integrating relational structures into pretrained seq2seq model for text-to-sql.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901. Association for Computational Linguistics.

Anuj Singh and Hadi Jamali-Rad. 2022. Transductive decoupled variational inference for few-shot classification.

Ningyuan Sun, Xuefeng Yang, and Yunfeng Liu. 2020. Tableqa: a large-scale chinese text-to-sql dataset for table-aware sql generation.

Ruoxi Sun, Sercan O. Arik, Hootan Nakhost, Hanjun Dai, Rajarishi Sinha, Pengcheng Yin, and Tomas Pfister. 2023. Sql-palm: Improved large language model adaptation for text-to-sql.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Chang-You Tai, Ziru Chen, Tianshu Zhang, Xiang Deng, and Huan Sun. 2023. Exploring chain-of-thought style prompting for text-to-sql.

Bailin Wang, Mirella Lapata, and Ivan Titov. 2021. Meta-learning for domain generalization in semantic parsing.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7567–7578. Association for Computational Linguistics.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020b. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Congying Xia, Caiming Xiong, Philip Yu, and Richard Socher. 2020. Composed variational natural language generation for few-shot intents. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3379–3388, Online. Association for Computational Linguistics.

Kuan Xu, Yongbo Wang, Yongliang Wang, Zihao Wang, Zujie Wen, and Yang Dong. 2022. SeaD: End-to-end text-to-SQL generation with schema-aware denoising. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1845–1853, Seattle, United States. Association for Computational Linguistics.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning.

Jingfeng Yang, Haoming Jiang, Qingyu Yin, Danqing Zhang, Bing Yin, and Diyi Yang. 2022. SEQZERO: Few-shot compositional semantic parsing with sequential prompts and zero-shot models. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 49–60, Seattle, United States. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association*

for Computational Linguistics, pages 8413–8426, Online. Association for Computational Linguistics.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. 2018. Structvae: Tree-structured latent variable models for semi-supervised semantic parsing.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2021. Grappa: Grammar-augmented pre-training for table semantic parsing.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander R. Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter S. Lasecki, and Dragomir R. Radev. 2019. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1962–1979. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium. Association for Computational Linguistics.

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

A Appendix

A.1 Data Statistics

Dataset	Train	Dev	Test
WikiSQL	56,355	8,421	15,878
ESQL	10,000	1,000	2,000

Table 5: Data statistics of the NL-SQL pairs.

Table 5 shows the data statistics of WikiSQL and ESQL. WikiSQL (Zhong et al., 2017) and ESQL (Chen et al., 2021). Both of them are single-table text-to-SQL datasets. The data statistics is shown in Table 5 in Appendix. WikiSQL is a significantly large dataset comprising over 20,000 open-domain tables sourced from Wikipedia. It stands as the most extensive single-table text-to-SQL dataset available to date. On the other hand, ESQL is a Chinese dataset that focuses on specific domains and consists of 17 tables.

A.2 Self Training

For self-training, a base model is first trained using a limited number of training samples. And then the base model is used to annotate large scale unlabeled dataset. Third, a confidence metric is designed and is used to select high-confidence samples. Fourth, the pseudo-samples are then added to the training set to re-train the base model. And this process can be repeated several times until the model converges. The challenges are three folds. First, a good base model is necessary. Second, a well-designed confidence metric is required. Third, we need to balance the ratio of pseudo-labeled and labeled data.

A.3 More introductions on the pretrained encoders

TAPAS (Herzig et al., 2020) initially designed for Table QA, also exhibits table encoding capabilities that can be applied to text-to-SQL tasks. TABERT (Yin et al., 2020) focuses on the joint understanding of textual and tabular data, incorporating table content into the representation. GRAPPA (Yu et al., 2021) combines table semantic parsing with a grammar-augmented pre-training framework. STRUG (Deng et al., 2021) proposes a set of structure-grounded pretraining tasks for effective text-table alignment. STAMP (Giaquinto et al., 2023) preforms large-scale multitask pretraining encoder-decoder models for interacting with structured knowledge.

A.4 Training settings

For training, we generally follow the settings of Guo et al. (2022) and run all the experiments on a platform with 8-V100 GPUs. The number of table content was set to $n = 5$. We only change the self-training module of MST-SQL and keep the column-specified meta learning unchanged. For fully supervised training on WikiSQL, we use the complete training set of WikiSQL as the labeled data $\mathcal{L}$ and employ WikiTableQuestions (Pasupat and Liang, 2015) as the unlabeled data $\mathcal{U}$, consisting of over 2,108 tables and 22,033 natural language queries (NLQs) without SQL labels.

For few-shot settings, we use the dataset created by Guo et al. (2022), which contains approximately 9,000 new tables in WikiSQL and 15 tables

Method	Dev.LF	Dev.EX	Test.LF	Test.EX
SQLOVA	81.6	87.2	80.7	86.2
X-SQL	83.8	89.5	83.3	88.7
HydraNet	83.6	89.1	83.8	89.2
SeaD	84.9	90.2	84.7	90.1
MC-SQL†	84.1	89.7	83.7	89.4
BRIDGE†	86.2	91.7	85.7	91.1
IE-SQL‡	84.6	88.7	84.6	88.8
SDSQL‡	86.0	91.8	85.6	91.4
TAPAS†‡	85.1	-	83.6	-
TABERT†‡	84.0	89.6	83.7	89.1
GRAPPA†‡	85.9	-	84.7	-
MST-SQL†‡	86.4	91.9	85.8	91.6
DeLVe-SQL†‡	**87.0**	**92.5**	**86.5**	**92.1**

Table 6: Fully supervised results on WikiSQL.† denotes the table-content is used and ‡ denotes the tabular pretrained method is used.

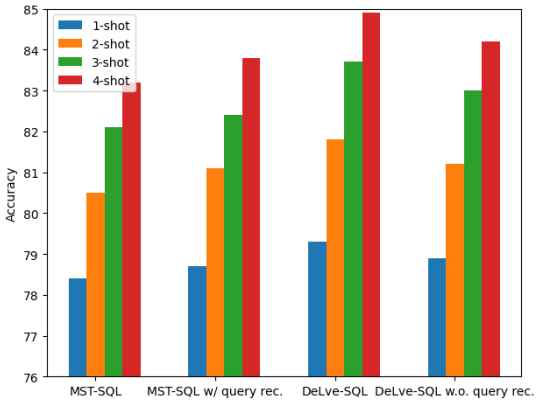

Figure 4: The effect of query reconstruction on WikiSQL test set. The accuracy value starting at the y-axis is 76.

in ESQL. {1, 2, 3, 4}-shot and and {5, 10, 15, 20}-shot for ESQL are constructed for WikiSQL and ESQL, respectively. The validation and test sets are constructed with the same settings, consisting of 4/100 samples for each table in WikiSQL and ESQL, respectively. For WikiSQL, the remaining NLQs from the original WikiSQL dataset is used to construct the unlabeled set $\mathcal{U}$.

A.5 Fully Supervised Results

Table 6 presents the fully supervised results on WikiSQL, divided into four distinct blocks.

Following the previous best performed model, namely MST-SQL (Guo et al., 2022), we compare our model with four kinds of recent approaches on the WikiSQL dataset. The first category includes SQLOVA (Hwang et al., 2019), X-SQL (He et al., 2019), HydraNet (Lyu et al., 2020) and SeaD (Xu et al., 2022), which did not utilize either table contents or tabular pretrained encoders. The second category includes BRIDGE (Lin et al., 2020) and MC-SQL (Chen et al., 2021), which employed table content understanding but did not utilize tabular pretraining. The third category contains IE-SQL (Ma et al., 2020) and SDSQL (Hui et al., 2021), which uses tabular pretraining but not table content. The fourth category which leverages both tabular pretraining and table contents includes TABERT (Yin et al., 2020), GRAPPA (Yu et al., 2021), TAPAS (Herzig et al., 2020), and MST-SQL (Guo et al., 2022).

In general, the performance of models that did not leverage table contents or tabular pretraining was inferior. The inclusion of either table contents or tabular pretraining significantly improved model performance, demonstrating the utility of these elements in generating more accurate SQLs. Our proposed model, DeLVe-SQL, achieves a LF score of 87.0% and an EX score of 92.5% in the Dev set, and a LF score of 86.5% and an EX score of 92.1% in the test set, outperforming the other methods including MST-SQL. The results validate the effectiveness of our proposed decoupled latent variable model in improving the generation of SQL queries in fully supervised training. The advantage of DeLVe-SQL can be attributed to its decoupled design, which allows it to align language semantics with database schema, and thereby improves the understanding of table structure. The use of latent variables facilitates fine-grained control over this alignment and offers an opportunity to disentangle the contribution of different parts of the input. Moreover, the optimization of these variables via language modeling loss further refines this alignment, thereby boosting performance.

A.6 Effect of Query Reconstruction

We examine the effectiveness of the query reconstruction module on both MST-SQL and DeLVe-SQL. For MST-SQL, the hidden vectors from the submodule classifier are directly fed to the GPT2 decoder without involving any latent variables. Figure 4 shows the results. In terms of MST-SQL, we can observe a slight improvement in performance when query reconstruction is applied. For example, the accuracy increases from 78.4% to 78.7% for 1-shot, and from 83.2% to 83.8% for 4-shot. This suggests that incorporating the query reconstruction module also helps enhance the accuracy of the MST-SQL model. For DeLVe-SQL, we can observe a decrease in performance when query re-

construction is removed. The accuracy decreases from 79.3% to 78.9% for 1-shot and from 84.9% to 84.2% without query reconstruction.

Overall, the results indicate that incorporating the query reconstruction module has a positive effect on the accuracy of both MST-SQL and DeLVe-SQL models. This suggests that the query reconstruction module helps improve the quality of generated SQL queries, leading to enhanced performance in the few-shot text-to-SQL task.

A.7 Significance of the Improvements.

We have done the significant test using the student's t-test against the baseline MSQ-SQL. In Table 7, all the results of DeLVe are significant at the p=0.05 level.

A.8 Model Variances.

Although the baseline MST-SQL does not show the variance scores, we additionally aly show the variance scores of 3 runs in Table 8. The variance numbers, being relatively low, indicate that the improvements brought by DeLVe-SQL are consistent and not just the result of random fluctuations or specificity ties of the dataset.

A.9 Quality of the Generated Query Text

Although the SQL-to-text utility is a byproduct. We thoroughly evaluate its text generation quality to show the capabilities of the reconstruction module. For the generated text, we evaluate them on the WikSQL and ESQL dev sets using BLEU-1, 2, 3, 4 scores. Table 9 shows the results. For both datasets, the BLEU scores gradually increase as the shot number increases. For WikiSQL, the scores remain relatively lower since at most 4 shots are used. For ESQL, the scores are higher than WikiSQL due to more data being available.

We also find some successful examples by reading the generated results. For example, when the original query is "How many people attended the game on May 10?", the generated query is "How many person attended the game on May 10?". Similarly, for the query "What is the sum of attendance when the score was 2:0?", the generated result is "What is the total attendance where the score ended as 2-0?", which can be seen as paraphrases.

A.10 More on Evaluation Metrics

Logical Form (LF) Accuracy measures how accurately the generated SQL query matches the correct SQL query in its structure and syntax. It doesn't consider whether the query returns the correct results when executed against a database. Instead, it focuses on the syntactical correctness and structural similarity of the SQL query to a reference query at the textual level. High LF accuracy indicates that the system is effective in parsing and structuring complex queries. The limitation is that a query might have correct syntax and structure (high LF accuracy) but still return incorrect results due to logical errors like incorrect conditions or joins.

Execution (EX) Accuracy assesses whether the SQL query, when executed, returns the correct data from the database. It's a measure of the query's functional correctness, regardless of its syntactic form. For example, given an example query "What is the total sales amount for 2022?". The golden query is "SELECT SUM(sales) FROM transactions WHERE year = 2022", suppose that the generated query is "SELECT SUM(sales) FROM transactions WHERE year < 2023 and year > 2021". For the generated query, the LF accuray is not 100%, since its syntax structure is different from the gold query. The EX accuracy is 100% since the results returned by the generated query are the same as the executed results of the gold query.

A.11 More on Comparisons with GPT-4

In this study, our method is specifically tailored to classical few-shot learning settings, which fundamentally differ from the prompt-based approaches used in large language models. While large models like GPT-4 can be potent for few-shot text-to-SQL tasks, there are several critical reasons why a direct comparison is not applicable in our context:

- Privacy Concerns: One of the foremost limitations is the issue of data privacy. In many cases, especially with private industry data, it's not permissible or secure to upload data to external platforms like GPT-4. Our methodology respects and upholds data privacy, an essential aspect for many industry applications.

- Data Exposure Uncertainty: There's an inherent uncertainty in large language models regarding prior exposure to training data. Since these models are often trained on vast, openly available datasets, it's unclear if they have already seen similar data, potentially skewing results and comparisons.

- Resource Intensity: The deployment and operation of large language models require sub-

	WikiSQL (#shots)				ESQL (#shots)			
Method	1	2	3	4	5	10	15	20
MST-SQL	78.4	80.5	82.1	83.2	55.3	67.4	76.7	80.5
DeLVe-SQL	79.3 (p=0.05)	81.8 (p=0.005)	83.7 (p=0.005)	84.9 (p=0.005)	56.8 (p=0.05)	68.9 (p=0.05)	78.7 (p=0.05)	82.0 (p=0.005)

Table 7: Significance Tests

	WikiSQL (#shots)				ESQL (#shots)			
Method	1	2	3	4	5	10	15	20
MST-SQL	78.4	80.5	82.1	83.2	55.3	67.4	76.7	80.5
DeLVe-SQL	79.3 (0.7)	81.8 (0.6)	83.7 (0.6)	84.9 (0.5)	56.8 (1.0)	68.9 (0.7)	78.7 (0.5)	82.0 (0.4)

Table 8: Model Variances

Dataset	**BLEU-1**	**BLEU-2**	**BLEU-3**	**BLEU-4**
WikiSQL 1-shot	0.60	0.42	0.32	0.20
WikiSQL 2-shot	0.65	0.48	0.38	0.28
WikiSQL 3-shot	0.69	0.52	0.42	0.32
WikiSQL 4-shot	0.74	0.59	0.49	0.39
ESQL 5-shot	0.82	0.67	0.57	0.47
ESQL 10-shot	0.85	0.70	0.60	0.50
ESQL 15-shot	0.86	0.73	0.67	0.57
ESQL 20-shot	0.88	0.77	0.70	0.63

Table 9: BLEU-scores for the reconstructed queries.

stantial computational resources. This high demand can pose significant challenges for practical, real-world applications, particularly in terms of deployment and ongoing service provision.

In summary, while large language models have their strengths, their application and comparison in a classical few-shot learning context are limited by privacy concerns, data exposure uncertainties, and high resource requirements. These factors make our approach more relevant and practical for certain scenarios, especially those prioritizing data privacy and computational efficiency.