PHOTON: A Robust Cross-Domain Text-to-SQL System

Jichuan Zeng[1*†] **Xi Victoria Lin**[2*] **Caiming Xiong**[2] **Richard Socher**[2]
Michael R. Lyu[1] **Irwin King**[1] **Steven C.H. Hoi**[2]
[1] The Chinese University of Hong Kong
[2] Salesforce Research
[1] {jczeng,lyu,king}@cse.cuhk.edu.hk
[2] {xilin,cxiong,rsocher,shoi}@salesforce.com

Abstract

Natural language interfaces to databases (NLIDB) democratize end user access to relational data. Due to fundamental differences between natural language communication and programming, it is common for end users to issue questions that are ambiguous to the system or fall outside the semantic scope of its underlying query language. We present PHOTON, a robust, modular, cross-domain NLIDB that can flag natural language input to which a SQL mapping cannot be immediately determined. PHOTON consists of a strong neural semantic parser (63.2% structure accuracy on the Spider dev benchmark), a human-in-the-loop question corrector, a SQL executor and a response generator. The question corrector is a discriminative neural sequence editor which detects confusion span(s) in the input question and suggests rephrasing until a translatable input is given by the user or a maximum number of iterations are conducted. Experiments on simulated data show that the proposed method effectively improves the robustness of text-to-SQL system against untranslatable user input. The live demo of our system is available at http://naturalsql.com/.

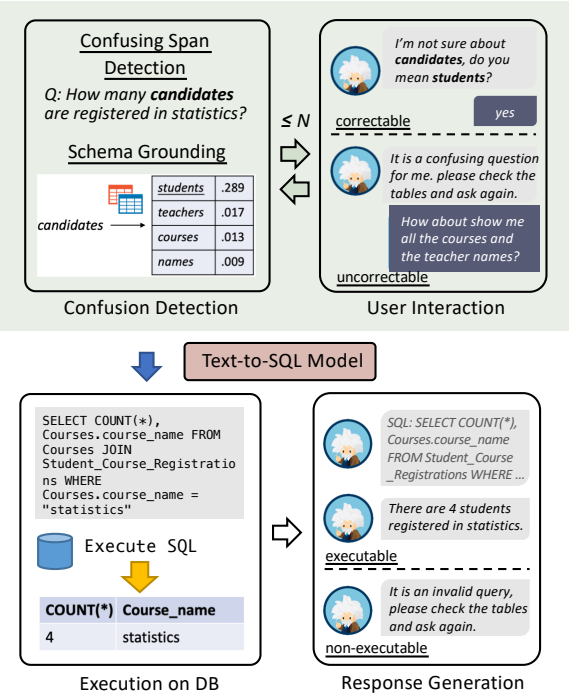


Figure 1: PHOTON workflow. The question corrector (upper block) detects the untranslatable questions from user input, scans the confusion span(s) that need clarification or correction. The accepted question is mapped into a SQL query through a text-to-SQL model, and finally the SQL execution results are returned to the user.

1 Introduction

Natural language interfaces to databases (Popescu et al., 2003; Li and Jagadish, 2014) democratize end user access to relational data and have attracted significant research attention for decades (Hemphill et al., 1990; Dahl et al., 1994; Zelle and Mooney, 1996; Popescu et al., 2003; Bertomeu et al., 2006; Zhong et al., 2017; Yu et al., 2018, 2019a). Most existing NLIDBs adopt a modular architecture consisting of rule-based natural language parsing, ambiguity detection and pragmatics modeling (Li and Jagadish, 2014; Setlur et al., 2016, 2019). While they have been shown effective in pilot study and production, rule-based approaches are limited in terms of coverage, scalability and naturalness – they are not robust against the diversity of human language expressions and are difficult to scale across domains.

Recent advances in neural natural language processing (Sutskever et al., 2014; Dong and Lapata, 2016; See et al., 2017a; Liang et al., 2017; Lin et al., 2019; Bogin et al., 2019a), pre-training (Devlin et al., 2019; Hwang et al., 2019), and the availability of large-scale supervised datasets (Zhong

[*] Equal contribution.
[†] This research was conducted during the author's internship at Salesforce Research.

et al., 2017; Finegan-Dollak et al., 2018; Yu et al., 2018, 2019b,a) enabled deep learning based approaches to significantly improve the state-of-the-art in nearly all subtasks of building an NLIDB. These include semantic parsing (Dong and Lapata, 2018; Zhang et al., 2019), ambiguity detection and confidence estimation (Dong et al., 2018; Yao et al., 2019), natural language response generation (Liu et al., 2019) and so on. Moreover, by jointly modeling the natural language question and database schema in the neural space, latest text-to-SQL semantic parsers can work cross domains (Yu et al., 2018; Zhang et al., 2019).

We present PHOTON, a modular, cross-domain NLIDB that adopts deep learning in its core components. PHOTON consists of (1) a neural semantic parser, (2) a human-in-the-loop question corrector, (3) a SQL query executor and (4) a natural language response generator. The *neural semantic parser* assumes limited DB content access due to data privacy concerns (§ 3.1). It employs a BERT-based (Devlin et al., 2019) DB schema-aware question encoder and a pointer-generator decoder (See et al., 2017a) with static SQL correctness check. It achieves competitive performance on the popular cross-domain text-to-SQL benchmark, Spider (Yu et al., 2018) (63.2% structure accuracy on the dev set based on the official evaluation).[1] The *question corrector* is a neural sequence editor which detects potential confusion span(s) in the input question and suggests possible corrections for the user to give feedback. When an input question is successfully translated into an executable SQL query, the *response generator* generates a natural language response conditioned on the output of the SQL *query executor*.

A pilot study with non-expert SQL users shows that the system effectively increases the flexibility of user's natural language expression and is easy to be adapted to unseen databases. Being able to detect and correct untranslatable questions reduces unexpected error cases during user interaction.

2 System Design

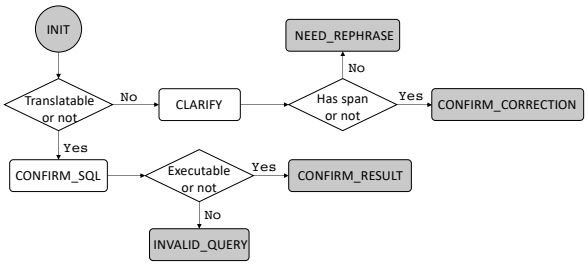
In this section, we will elaborate on the system design of PHOTON.

[1]We are continuously improving the performance of the neural semantic parser. Currently the semantic parser only accepts standalone question as input. We plan to also model the interaction context in future work.

2.1 Overview

Figure 1 shows the overall workflow of our system. PHOTON is an end-to-end system that takes a user question and database schema as input, and output the query result after executing the generated SQL on the database. PHOTON is a modular framework designed towards practical industrial applications. The core modules in PHOTON are the SQL parser and confusion detection mechanism. The SQL parser parses the input question and database schema, maps them into executable SQL query via an encoder-decoder framework. The confusion detection module identifies the untranslatable questions and captures the confusing span of the untranslatable question. The confusing tokens together with the context are fed into the auto-correction module to make a prediction of user attempted question.

To make it more applicable and accessible for user to query the database in a natural way, PHOTON also provides user interaction module enabling user to refine their queries in the interaction with the system. Response generation handles the output of the system by transducing the database-style query result into natural language or post warning when the query is non-executable on the database, making the system more user-friendly. Notice that the response generation module in the current version is implemented using a template-based approach and can be improved by using more advanced response generation models.



Response Template

CONFIRM_RESULT	"SQL: {PRED_SQL}. {NL_RESPONSE}"
CONFIRM_CORRECTION	"Sorry, {CONF_TOKENS} is confusing in our scenario, do you mean {CORR_TOKENS}?"
NEED_REPHRASE	"Sorry, it is a confusing question for me, please rephrase your question and ask again."
INVALID_QUERY	"Sorry, it is an invalidate query, please check the table names and associated fields of interest."

Figure 2: State transition map of interaction in PHOTON. States with darker background are the end states that can receive user reply, and switch to `INIT` state automatically. The bottom part is the system response templates in each end state.

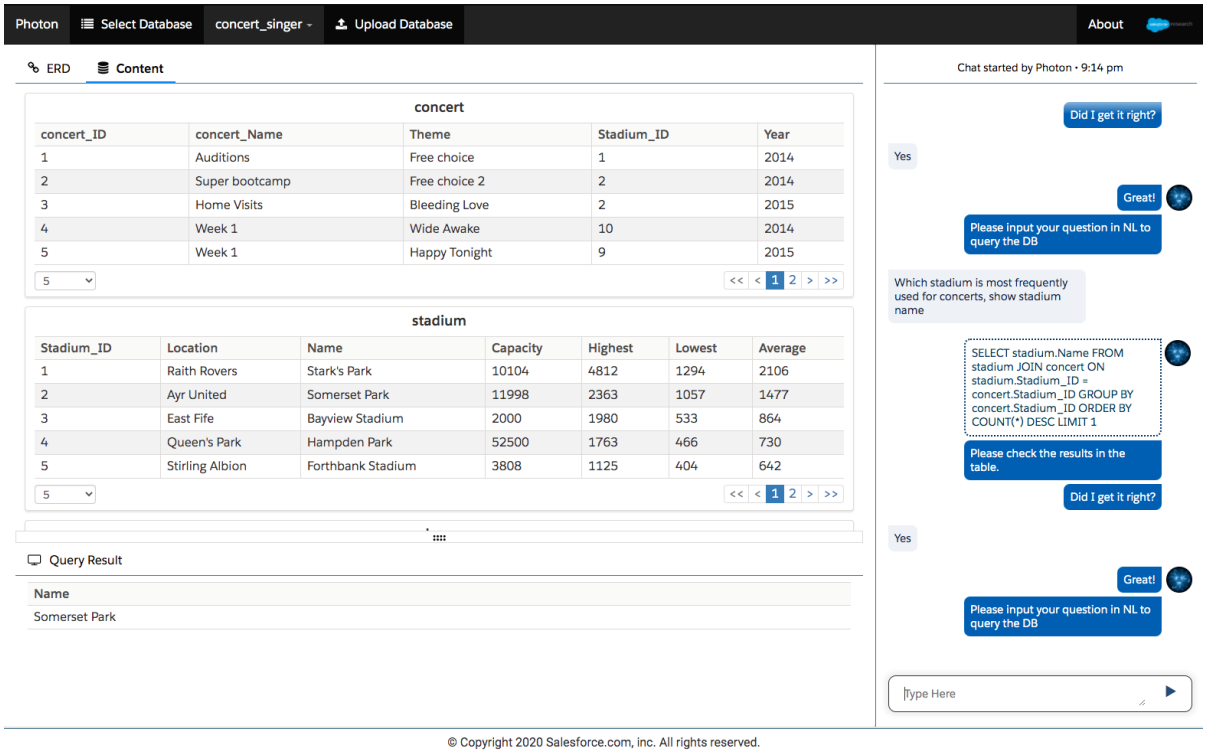


Figure 3: PHOTON main interface.

2.2 User Interaction

Figure 2 illustrates the interaction process, which involves four types of response states: CONFIRM_RESULT, CONFIRM_CORRECTION, NEED_REPHRASE, and INVALID_QUERY. The set of response templates can be found at the bottom of Figure 2. When a user initiates the conversation by entering one query, PHOTON will first predict whether the query is translatable or not. If translatable, PHOTON generates the corresponding SQL command and checks the command's executability; otherwise, PHOTON will provide a correction strategy (i.e., CONFIRM_CORRECTION) based on the detected confusing span or ask the users to further rephrase the inquiry (i.e., NEED_REPHRASE) if no span is captured.

2.3 UI Design

Our system UI consists of three panels: chat window, schema viewer and results viewer (Figure 3).

- *Chat window*: This is a standard chat window that facilitates communication between the user and PHOTON. The user types the natural language input and the natural language responses of the system are displayed.
- *Schema viewer*: This view provides a graph visualization of the underlying relational DB schema. The panel is hideable and will not be shown in case the DB schema is confidential.
- *Result viewer*: This view displays the returned results of an executable SQL query mapped from a confirmed input question. The SQL query is formatted and displayed in the top for user verification. Multi-record results are presented as sub-tables. Result consists of a single table cell is presented as a 1-cell subtable. If the result comes from an aggregation operation such as a counting, the data records supporting the calculation are also shown for explanability. Confidential DB records are hidden from the display and the user is informed of the number of hidden records.

2.4 Cross Domain

A relational DB for user queries should be set before usage. PHOTON consists of a collection of default databases and allows users to upload their own DBs for testing. Users can select which database they want to query by clicking the "Selected Database" drop down button.

2.5 Dual Query Mode

PHOTON accepts both natural language question and well-formed SQL queries as input. When the

user types an input, PHOTON automatically detects the input type. We observed that allowing the dual query model can be more efficient than only supporting natural language queries, especially for users who have SQL background.

3 Model

3.1 Neural Semantic Parser

The neural semantic parser is an end-to-end model whose input consists of a user question and the DB schema, and outputs a SQL query. Due to data privacy concerns, we assume that the neural semantic parser does not have full access to the DB content. Instead, we assume for each DB field, the parser have access to the set of possible values of the field, for example, "Country.Region": {"Carribean", "Porto Rico", ...}[2]. We call such value sets "picklists" by industry convention.

3.1.1 Schema-Question Encoder

Following previous work (Hwang et al., 2019; Zhang et al., 2019), we serialize the relational DB schema and concatenate it to the user question. As shown in Figure 4 , we represent each table with the table name followed by a sequence of field names. Each table name is preceded by the special token `[T]` and each field name is preceded by the special token `[C]`. The representations of multiple tables are concatenated together to form the serialization of the schema, which is surrounded by `[SEP]` tokens and concatenated to the question. Finally, the question is preceded by the `[CLS]` token following convention of BERT encoder (Devlin et al., 2019).

This sequence is fed into the pretrained BERT, followed by a bi-directional LSTM to form a joint encoding of the question and schema $\mathbf{h}_{\text{input}}$. The text portion of $\mathbf{h}_{\text{input}}$ is passed through another bi-LSTM to obtain the question encoding $\mathbf{h}_{\text{Q}}$. We represent each schema component (tables and fields) using the slices of $\mathbf{h}_{\text{input}}$ corresponding to the special token `[T]` and `[C]`.

Meta-data Features We further trained dense look-up features to represent if a field is a primary key ($\mathbf{f}_{\text{pri}}$), if a field appears in a foreign key pair ($\mathbf{f}_{\text{for}}$) and the data type of the field ($\mathbf{f}_{\text{type}}$). These meta-data features are fused with the representations in $\mathbf{h}_{\text{input}}$ via a projection layer g to obtain the final representation of each schema component:

$$\mathbf{h}^{C_p} = g([\mathbf{h}^m_{\text{input}}; \mathbf{f}^i_{\text{pri}}; \mathbf{f}^j_{\text{for}}; \mathbf{f}^k_{\text{type}}]) \quad (1)$$
$$= \text{ReLU}(\mathbf{W}_g[\mathbf{h}^m_{\text{input}}; \mathbf{f}^i_{\text{pri}}; \mathbf{f}^j_{\text{for}}; \mathbf{f}^k_{\text{type}}] + \mathbf{b}_g)$$
$$\mathbf{h}^{T_q} = g([\mathbf{h}^n_{\text{input}}; \mathbf{0}; \mathbf{0}; \mathbf{0}]), \quad (2)$$

where m is the index of the special token corresponding to the p-th column in the input and n is the index of the special token corresponding to the q-th table in the input. i, j and k are the feature indices indicating the corresponding properties of C_p. $[\mathbf{h}^m_{\text{input}}; \mathbf{f}^i_{\text{pri}}; \mathbf{f}^j_{\text{for}}; \mathbf{f}^k_{\text{type}}]$ is the concatenation of the four vectors. The meta-data features we include are specific to fields and the table representations are fused with zero place-holder vectors.

3.1.2 Decoder

We use an LSTM-based sequential pointer-generator (See et al., 2017b) as the decoder. The generation vocabulary of our decoder consists of 70 SQL keywords and reserved tokens, plus the 10 digits[3]. At each step, the decoder computes a probability distribution over actions that consists of generating a token from the reserved vocabulary, copying a token from the input text or copying a schema component.

3.1.3 Static SQL Correctness Check

The sequential pointer-generator we adopted does not guarantee the output SQL is syntactically correct. In practice, we perform beam-search decoding and run a static SQL correctness check[4] to eliminate erroneous predictions from the beam. Specifically, we employ a tool implemented on top of the Mozilla SQL Parser[5] to analyze the output SQL queries and ensure they satisfy the following criteria:

1. The SQL query is syntactically correct.
2. The SQL query satisfies schema consistency[6].

We found this approach is very effective and results in an absolute improvement of 4∼5% in the evaluation score on Spider dev set (Yu et al., 2018).

[2] In practice, we can limit the access to only certain fields.

[3] Such that the parser is able to generate numbers corresponding to utterances such as "first", "second" etc.

[4] Some prior work such as (Wang et al., 2018) performs a similar check by executing the decoded SQL queries on the target DB. We implement the static checking as it can reduce the traffic between the interface and the DB.

[5] https://github.com/mozilla/moz-sql-parser

[6] The fields appeared in a `SELECT` SQL query must come from the tables in the corresponding `FROM` clause. The fields in a `JOIN` condition clause must come from tables mentioned in front of them in the `JOIN` clause.

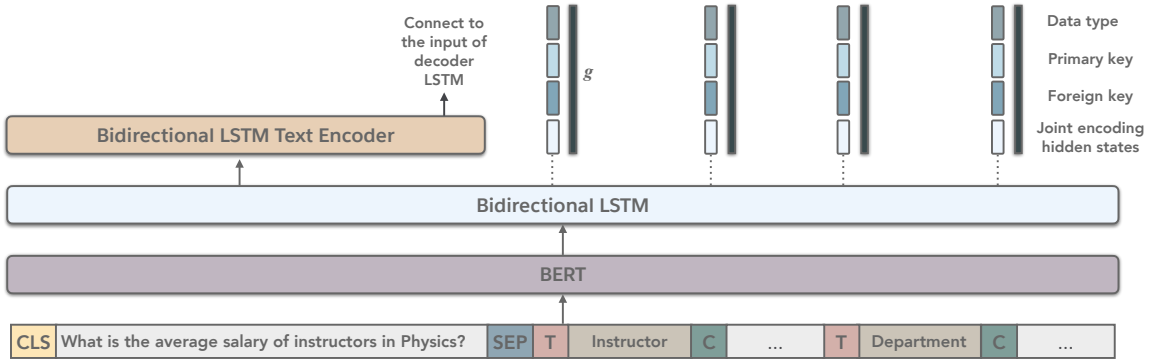


Figure 4: Joint schema-question encoder.

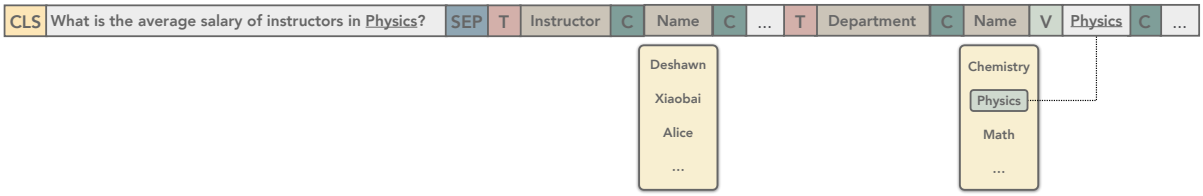


Figure 5: Joint schema-question encoder with picklist value augmentation.

3.1.4 Picklist Incorporation

We use picklists to inform the semantic parser regarding potential matches in the DB. For an input question Q and a field C_p, we compute the longest character sequence match between Q and each value in the picklist of C_p. We select the value with top-1 matching score above a certain threshold θ as a match. For each field with a matched picklist value, we append the surface form of the value to it in the input sequence representation, separated by the special token `[V]`. The augmented sequence is used as the input to the schema-question encoder. In practice, we found picklist augmentation results in an absolute performance improvement of 1% on the Spider dev set.

Figure 5 illustrates the input sequence with augmented picklist values. In this example, the matching algorithm identifies "Carribean" associated with the column "Country.Region" as a match. Hence it inserts "Carribean" after [... `[C]`, "Region"] with `[V]` as a separation token[7]. The representations of fields with no picklist value match are unchanged.

[7]In practice, we found a question typically has 0 to 4 picklist value matches. As a result, the picklist augmented schema-question representation still stays under the maximum input length of BERT.

3.2 Confusion Detection: Handling Untranslatable and Ambiguous Input

In order to handle ambiguous and untranslatable input questions, PHOTON adopts a discriminatively trained classifier to detect user input to which a SQL mapping cannot be immediately determined. This covers questions that are incomplete (e.g. *What is the total?*), ambiguous or vague (e.g. *Show me homes with good schools*), beyond the representation scope of SQL (e.g. *How many tourists visited all of the 10 attractions?*), or simply noisy (e.g. *Cyrus teaches physics in department*).

3.2.1 Untranslatable Question Detection

Inspired by (Rajpurkar et al., 2018), we create a synthetic dataset which consists of untranslatable questions generated by applying rule-based transformations and adversarial filtering (Zellers et al., 2018) to examples in existing text-to-SQL datasets. We then train a stagewise model that first classifies if the input is translatable or not, and then predicts confusing spans in an untranslatable input.

Dataset Construction. In order to construct the untranslatable questions, we firstly exam the types of untranslatable questions seen on the manually constructed CoSQL (Yu et al., 2019a) and MultiWOZ (Budzianowski et al., 2018) datasets (Table 4 of A.1). We then design our modification strategies to generate the untranslatable questions from the original text-to-SQL dataset automatically. Specif-

ically, for a text-to-SQL example that contains a natural language question, a DB schema and a SQL query, we first identify all non-overlapping question spans that possibly refer to a table field occurred in the `SELECT` and `WHERE` clauses of the SQL query using string-matching heuristics. Then we apply *Swap* and *Drop* operations on the question and DB schema respectively to generate different types of untranslatable questions. The modification tokens are marked as the confusion spans of the synthetic untranslatable questions, except for the question *Drop* strategy.

Table 5 in A.1 provides a detailed summary of all transformations applied[8]. For example, given the original question "How many countries exist?", "countries" is detected to be referring to a table field. We drop the token, and pass the modified question "How many exist?" to back-translation for grammar smoothing. After that, we obtain the untranslatable question "How many are there?". Once we have the synthetic untranslatable questions, adversarial filtering is employed to iteratively refine the set of untranslatable examples to be more indiscernible by trivial stylistic classifiers (Zellers et al., 2018).

Predicting Untranslatable Questions and Confusing Spans. We utilize the BERT-based contextualized question-schema representations produced by the encoder shown in Figure 5 to extract the confusion span in a question hence predict its translatability. To do so, we normalize[9] aforementioned synthetic dataset such that each question Q in an example is paired with a span label ζ, where

$$\zeta = \begin{cases} (s+1, t+1), & \text{if } Q \text{ is untranslatable with confusion span } (s,t) \\ (1, |Q|+1), & \text{if } Q \text{ is untranslatable with no identifiable confusion span} \\ (0,0). & \text{if } Q \text{ is translatable} \end{cases} \quad (3)$$

We use the span extractor as the one proposed by Devlin et al. (2019) for SQuAD v1.1 (Rajpurkar et al., 2016). Specifically, the extractor introduces a start vector $\mathbf{s}$ and an end vector $\mathbf{e}$. The probability of word i being the start of the answer span is conputed as a dot product between $\mathbf{h}^i_{\text{input}}$ and $\mathbf{s}$ followed by a softmax over `[CLS]` and all words in the question. The end of the answer span is predicted in a similar manner.

$$p^i_{\text{start}} = \frac{e^{\mathbf{s}^\top \mathbf{h}^i_{\text{input}}}}{\sum_j e^{\mathbf{s}^\top \mathbf{h}^j_{\text{input}}}}, \; p^i_{\text{end}} = \frac{e^{\mathbf{e}^\top \mathbf{h}^i_{\text{input}}}}{\sum_j e^{\mathbf{e}^\top \mathbf{h}^j_{\text{input}}}}. \quad (4)$$

The score of a candidate span (i, j) is $\mathbf{s}^\top \mathbf{h}^i_{\text{input}} + \mathbf{e}^\top \mathbf{h}^j_{\text{input}}$, and the maximum scoring span with $j \geq i$ is used as a prediction. The training objective is the sum of the log-likelihood of the correct start and end indices.

3.2.2 Database-aware Token Correction

Figure 6 illustrates the proposed tokens correction module in PHOTON. We use the masked language model (MLM) of BERT (Devlin et al., 2019) to auto-correct the confusing tokens. Specifically, we replace the confusing tokens with the `[MASK]` special token. The output distribution of MLM head on the mask token is employed to score the candidate spans. We construct the candidate span list by extracting all the table names and columns names from the database schema. After user confirmation, the confusing tokens in the input are replaced by the predicted tokens of MLM.

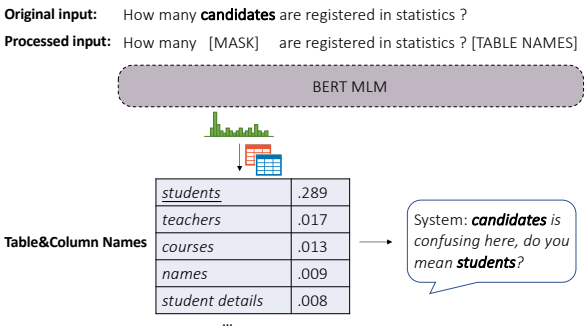


Figure 6: Token Correction in PHOTON.

4 Evaluation

In this section, we empirically evaluate the robustness and effectiveness of PHOTON.

In particular, we examine two key modules of PHOTON: the confusion detection module and the neural semantic parser. The former aims to detect the untranslatable questions and predicts the confusing spans; if the question is translatable, it then applies the proposed neural semantic parser to perform the text-to-SQL parsing. Since PHOTON is designed as a stagewise system, we can evaluate the performance of each module separately.

[8] To introduce semantic variation and ensure grammar fluency, we apply back-translation on the generated question using Google Cloud Translation API https://cloud.google.com/translate/. We use Chinese as the intermediate language.

[9] Essentially, when the question is untranslatable but there is no identifiable confusion span, the whole question is marked as confusing; when the question is translatable, `[CLS]` is used as the dummy confusion span.

4.1 Experimental Setup

Dataset. We conduct experiments on Spider (Yu et al., 2018) and $\text{Spider}_{\text{UTran}}$ dataset. Spider is a large-scale, human annotated, cross-domain text-to-SQL benchmark. $\text{Spider}_{\text{UTran}}$ is our modified dataset to evaluate robustness, created by injecting the untranslatable questions into Spider. We obtained 5,330 additional untranslatable questions (4,733 for training and 597 for development) from the original Spider dataset. To ensure the quality of our synthetic dataset, we hired SQL experts from Upwork[10] to annotate the auto-generated untranslatable examples in the dev set. We conduct our evaluation by following the database split setting, as illustrated in Table 1. The split follows the original dataset hence there is no test set of $\text{Spider}_{\text{UTran}}$ (the test set of Spider is not publicly accessible).

	Spider		**$\text{Spider}_{\text{UTran}}$**	
	Train	Dev	Train	Dev
# Q	8,659	1,034	13,392	1,631
# UTran Q	0	0	4,733	597
# Schema	146	20	918	112

Table 1: Data split of Spider and $\text{Spider}_{\text{UTran}}$. Q represents the all the questions, UTran Q represents the untranslatable questions.

Training and Inference Details. Our neural semantic parser is trained on Spider. We permute table order (up to 16 different ones) during training. We use the uncased BERT-base model from Huggingface's transformer library (Wolf et al., 2019). We set all LSTMs to 1-layer and set the dimension of $\mathbf{h}_{\text{input}}$, $\mathbf{f}_{\text{pri}}$, $\mathbf{f}_{\text{for}}$, $\mathbf{f}_{\text{type}}$ and the decoder to 512. We employ Adam-SGD (Kingma and Ba, 2015) with a mini-batch size of 32 and default Adam parameters. We train a maximum of 50,000 steps and set the learning rate to $5e-4$ in the first 5,000 iterations and linearly decays it to 0 afterwards. We fine-tune BERT with a fine-tuning rate linearly increasing from $3e-5$ to $8e-5$ in the first 5,000 iterations and linearly decaying to 0 afterwards. We use a beam size of 128 in the beam search decoding.

Our confusion span extractor is trained on $\text{Spider}_{\text{UTran}}$. It uses the same encoder architecture as the semantic parser. We set the dimension of $\mathbf{s}$ and $\mathbf{e}$ to 512. During inference, we determine a question as untranslatable if the predicted span start $s > 0$. PHOTON highlights the confusion span to the user when it contains 5 or less tokens; otherwise, it generates a generic response prompting the user to rephrase.

[10] https://www.upwork.com/

4.2 Experimental Results

Confusion Detection. We examine the robustness of PHOTON by evaluating the performance of the Confusion Detection module in handling ambiguous and untranslatable input. In particular, we aim to examine if PHOTON is effective in handling untranslatable questions by measuring the translatability detection accuracy and the confusing span prediction accuracy & F1 score[11]. We compare to a baseline that uses a single-layer attentive bidirectional LSTM ("Att-biLSTM"). Table 2 shows the evaluation results on the $\text{Spider}_{\text{UTran}}$ dataseet.

	Tran Acc	**Span Acc**	**Span F1**
Att-biLSTM	66.6	58.7	59.2
PHOTON	**87.6**	**83.6**	**84.7**

Table 2: Translatability prediction accuracy ("Tran Acc") and the confusing spans prediction accuracy and F1 on our $\text{Spider}_{\text{UTran}}$ dataset (%).

As observed from Table 2, PHOTON achieves encouraging performance in determining the translatability of a question and predicting the confusing spans of untranslatable ones. In comparison to the Att-biLSTM baseline, PHOTON obtains significant improvements in both translatability accuracy and the confusing spans prediction accuracy. These improvements are partly attribute to the proposed effective schema encoding strategy.

Neural Semantic Parser. We then evaluate the performance of the proposed neural semantic parser of PHOTON on the original Spider dataset. In particular, we compare PHOTON and other existing text-to-SQL approaches by measuring the exact set match (EM) accuracy (Yu et al., 2018). We compare with several existing approaches, including Global GNN (Bogin et al., 2019b), EditSQL (Zhang et al., 2019), IRNet (Guo et al., 2019), and RYANSQL (Choi et al., 2020). Table 3 shows the evaluation results on Spider Dev set.

As observed from Table 3, PHOTON achieves a very competitive text-to-SQL performance on the Spider benchmark with 63.2% exact set match accuracy on the Spider dev set, which validates the ef-

[11] We use the same way as SQuAD 2.0 (Rajpurkar et al., 2018) to compute the span accuracy and F1.

Model	EM Acc.
GNN (Bogin et al., 2019a)	40.7
Global-GNN (Bogin et al., 2019b)	52.7
EditSQL + BERT (Zhang et al., 2019)	57.6
GNN+Bertrand-DR† (Kelkar et al., 2020)	57.9
EditSQL+Bertrand-DR† (Kelkar et al., 2020)	58.5
IRNet + BERT (Guo et al., 2019)	61.9
RYANSQL + BERT † (Choi et al., 2020)	66.6
PHOTON	63.2

† denotes unpublished work on arXiv.

Table 3: Experimental results on the Spider Dev set (%). EM Acc. denotes the exact set match accuracy.

fectiveness of our neural semantic parser for translating an input question into a valid SQL query.

5 Related Work

Natural Language Interfaces to Databases. NLIDBs has been studied extensively in the past decades. Thanks to the availability of large-scale datasets (Zhong et al., 2017; Finegan-Dollak et al., 2018; Yu et al., 2018), data-driven approaches have dominated the field, in which deep learning based models achieve the best performance in both strongly (Hwang et al., 2019; Zhang et al., 2019; Guo et al., 2019) and weakly (Liang et al., 2017; Min et al., 2019) supervised settings. However, most of existing text-to-SQL datasets include only questions that can be translated into a valid SQL query. Spider (Finegan-Dollak et al., 2018) specifically controlled question clarify during data collection to exclude poorly phrased and ambiguous questions. WikiSQL (Zhong et al., 2017) was constructed on top of manually written synchronous grammars, and the mapping between its questions and SQL queries can be effectively resolved via lexical matching in vector space (Hwang et al., 2019). CoSQL (Yu et al., 2019a) is by far the only existing corpus to our knowledge which entables data-driven modeling and evaluation of untranslatable question detection. Yet the dataset is of context-dependent nature and contains untranslatable questions of limited variety. We fill in this gap by proposing PHOTON to cover a diverse set of untranslatable user input in text-to-SQL.

Noisy User Input in Semantic Parsing. Despite being absent from most large-scale text-to-SQL benchmarks, noisy user input has been frequently encountered and battled with by the semantic parsing community. Underspecification (Archangeli, 1988) and vagueness (Varzi, 2001) have solid linguistic theory foundation. Lexicon-based semantic parsers (Zettlemoyer and Collins, 2005; Roberts and Patra, 2017) may reject the input if the lexicon match is unsuccessful. Other approaches for handling untranslatable user input include inference and generating defaults (Setlur et al., 2019), paraphrasing (Arthur et al., 2015, 2016), verification (Arthur et al., 2015) and confidence estimation (Dong et al., 2018). We adopt a data-augmentation and discriminative learning based approach, which has demonstrated superior performance in related domains (Rajpurkar et al., 2018)

6 Conclusion and Future Work

We present PHOTON, a robust modular cross-domain text-to-SQL system, consisting of semantic parser, untranslatable question detector, human-in-the-loop question corrector, and natural language response generator. PHOTON has the potential to scale up to hundreds of different domains. It is the first cross-domain text-to-SQL system designed towards industrial applications with rich features, and bridges the demand of sophisticated database analysis and people without any SQL background knowledge.

The current PHOTON system is still a prototype, with very limited user interactions and functions. We will continue to add more features to PHOTON, such as voice input, spelling checking, and visualizing the output when appropriate to inspect the translation process. We also plan to improve the performance of core models in PHOTON, such as semantic parsing (text-to-SQL), response generation (table-to-text) and context-aware user interaction (text-to-text). A comprehensive evaluation will also be conducted among the users of our system.

References

Diana Archangeli. 1988. Aspects of underspecification theory. *Phonology*, 5(2):183–207.

Philip Arthur, Graham Neubig, Sakriani Sakti, and Satoshi Nakamura. 2016. Semantic parsing of ambiguous input using multi synchronous grammars.

Philip Arthur, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Semantic parsing of ambiguous input through paraphrasing and verification. *TACL*, 3:571–584.

Núria Bertomeu, Hans Uszkoreit, Anette Frank, Hans-Ulrich Krieger, and Brigitte Jörg. 2006. Contextual

phenomena and thematic relations in database QA dialogues: results from a wizard-of-Oz experiment. In *Proceedings of the Interactive Question Answering Workshop at HLT-NAACL 2006*, pages 1–8, New York, NY, USA. Association for Computational Linguistics.

Ben Bogin, Jonathan Berant, and Matt Gardner. 2019a. Representing schema structure with graph neural networks for text-to-sql parsing. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Vol. 1*, pages 4560–4565.

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019b. Global reasoning over database structures for text-to-sql parsing. *CoRR*, abs/1908.11214.

Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz - A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 5016–5026.

DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2020. RYANSQL: recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *CoRR*, abs/2004.03125.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William M. Fisher, Kate Hunicke-Smith, David S. Pallett, Christine Pao, Alexander I. Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology, Proceedings of a Workshop held at Plainsboro, New Jerey, USA, March 8-11, 1994*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1*, pages 4171–4186.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer 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, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 731–742.

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. In (Gurevych and Miyao, 2018), pages 743–753.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir R. Radev. 2018. Improving text-to-sql evaluation methodology. In (Gurevych and Miyao, 2018), pages 351–360.

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 *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4524–4535.

Iryna Gurevych and Yusuke Miyao, editors. 2018. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*. Association for Computational Linguistics.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, USA, June 24-27, 1990*.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *CoRR*, abs/1902.01069.

Amol Kelkar, Rohan Relan, Vaishali Bhardwaj, Saurabh Vaichal, and Peter Relan. 2020. Bertrand-dr: Improving text-to-sql using a discriminative re-ranker. *arXiv preprint arXiv:2002.00557*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Anna Korhonen, David R. Traum, and Lluís Màrquez, editors. 2019. *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Association for Computational Linguistics.

Fei Li and Hosagrahar Visvesvaraya Jagadish. 2014. Nalir: an interactive natural language interface for querying relational databases. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 709–712. ACM.

Chen Liang, Jonathan Berant, Quoc V. Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational*

Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers, pages 23–33. Association for Computational Linguistics.

Kevin Lin, Ben Bogin, Mark Neumann, Jonathan Berant, and Matt Gardner. 2019. Grammar-based neural text-to-sql generation. *CoRR*, abs/1905.13326.

Tianyu Liu, Fuli Luo, Pengcheng Yang, Wei Wu, Baobao Chang, and Zhifang Sui. 2019. Towards comprehensive description generation from factual attribute-value tables. In (Korhonen et al., 2019), pages 5985–5996.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. A discrete hard EM approach for weakly supervised question answering. *CoRR*, abs/1909.04849.

Ana-Maria Popescu, Oren Etzioni, and Henry A. Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces, IUI 2003, Miami, FL, USA, January 12-15, 2003*, pages 149–157. ACM.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *ArXiv*, abs/1606.05250.

Kirk Roberts and Braja Gopal Patra. 2017. A semantic parsing method for mapping clinical questions to logical forms. In *AMIA 2017, American Medical Informatics Association Annual Symposium, Washington, DC, USA, November 4-8, 2017*. AMIA.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017a. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017b. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Vidya Setlur, Sarah E. Battersby, Melanie Tory, Rich Gossweiler, and Angel X. Chang. 2016. Eviza: A natural language interface for visual analysis. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology, UIST 2016, Tokyo, Japan, October 16-19, 2016*, pages 365–377. ACM.

Vidya Setlur, Melanie Tory, and Alex Djalali. 2019. Inferencing underspecified natural language utterances in visual analysis. In *Proceedings of the 24th International Conference on Intelligent User Interfaces, IUI 2019, Marina del Ray, CA, USA, March 17-20, 2019*, pages 40–51. ACM.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Achille C Varzi. 2001. Vagueness, logic, and ontology.

Chenglong Wang, Po-Sen Huang, Alex Polozov, Marc Brockschmidt, and Rishabh Singh. 2018. Execution-guided neural program decoding. *CoRR*, abs/1807.03100.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Ziyu Yao, Yu Su, Huan Sun, and Wen-tau Yih. 2019. Model-based interactive semantic parsing: A unified framework and A text-to-sql case study. 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 5446–5457. Association for Computational Linguistics.

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 Richard Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter S. Lasecki, and Dragomir R. Radev. 2019a. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. *CoRR*, abs/1909.05378.

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. 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, Oct 31 - Nov 4, 2018*, pages 3911–3921.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir R. Radev. 2019b. Sparc: Cross-domain

semantic parsing in context. In (Korhonen et al., 2019), pages 4511–4523.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 2.*, pages 1050–1055.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 93–104. Association for Computational Linguistics.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 658–666. AUAI Press.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir R. Radev. 2019. Editing-based SQL query generation for cross-domain context-dependent questions. *CoRR*, abs/1909.00786.

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

A Appendices

A.1 Construct Untranslatable Questions

Table 4 shows a summary of different types of untranslatable questions based on analysis of CoSQL (Yu et al., 2019a) and Multi-WOZ (Budzianowski et al., 2018).

Table 5 shows examples of applying question-side and schema-side transformations to convert a translatable question from existing text-to-SQL datasets to an untranslatable question.

Reason	Description	Example
Underspecification	Input does not specify which data entries/attributes to query.	Q: What is the total? Schema:||Course_ID||Staring_Data||Course||…
Overspecification	Input asks for information that cannot be found in the DB.	Q: What is the name of the singer with the largest **net worth**? Schema: ||Singer_ID||Name||Birth_Year||Citizenship||
Ambiguity & Vagueness	Input contains ambiguous or vague expressions.	Q: Show me homes with **good** schools Schema: ||Address||Community||School Name||School Rating||
Beyond representation scope of SQL	Input asks for information that cannot be obtained by SQL logic.	Q: What is the **trend** of housing price this year? Schema: ||House ID||Location||Price||Number of amenties||
Not a query	Input is not a linguistically valid question.	Q: Cyrus teaches physics in department.
Others	Other cases that the question cannot be translated.	Q: How many **Russias** have Summer's transfer window? Schema: ||Name||Country||Type||Transfer Window||Transfer Fee||

Table 4: Types of untranslatable questions in text-to-SQL identified from manual analysis of CoSQL (Yu et al., 2019a) and Multi-WOZ (Budzianowski et al., 2018). A question span that is problematic for the translation is highlighted when applicable.

Transformation		Original data	Transformed data	Confusing text span
Question	Swap	Q1: How many *conductors* are there? S1: || Conductor_ID || Name || Age || Nationlity|| Year_of_Work ||	Q1: How many **soloists** are there ?	soloists
		Q2: What are the maximum and minimum values of *area codes* ? S2: || Vote_ID || Phone_Number || Area_Code || State || Created ||	Q2: What are the maximum and minimum values of **types**?	types
	Drop	Q1: How many *countries* exist? S1: || CoutryId || CountryName || Continent ||	Q1: How many are there?	WHOLE SENTENCE
		Q2: What is the *official language* spoken in the country whose head of state is Beatrix? S2: || CountryCode || HeadOfState || Captital || Language || IsOfficial || Percentage ||	Q2: What are the people in the country where Beatrix is located?	WHOLE SENTENCE
Schema Drop		Q1: How much *surface area* do the countires in the Carribean cover together? S1: || Name ||Continent || Region || *SurfaceArea* || Population || LifeExpectancy ||	S1: || Name ||Continent || Region || Population || LifeExpectancy ||	surface area
		Q2: Find the name and *age* of the visitor who bought the most tickets at once. S2: ||Customer_ID||Name||Level_of_membership|| *Age*||	S2: ||Customer_ID||Name||Level_of_membership||	age

Table 5: Examples of question-side and schema-side transformations for generating training data for untranslatable question detection. Let Q denote the question and S denote the schema. For each transformation, we provide two examples, i.e., (Q1, S1) and (Q2, S2). The italic and bold fonts highlight phrases before and after transformations.