Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions

Abstract

We focus on the cross-domain context-dependent text-to-SQL generation task. Based on the observation that adjacent natural language questions are often linguistically dependent and their corresponding SQL queries tend to overlap, we utilize the interaction history by editing the previous predicted query to improve the generation quality. Our editing mechanism views SQL as sequences and reuses generation results at the token level in a simple manner. It is flexible to change individual tokens and robust to error propagation. Furthermore, to deal with complex table structures in different domains, we employ an utterance-table encoder and a table-aware decoder to incorporate the context of the user utterance and the table schema. We evaluate our approach on the SParC dataset and demonstrate the benefit of editing compared with the state-of-the-art baselines which generate SQL from scratch. Our code is available at this https URL.

Introduction

Generating SQL queries from user utterances is an important task to help end users acquire information from databases. In a real-world application, users often access information in a multi-turn interaction with the system by asking a sequence of related questions. As the interaction proceeds, the user often makes reference to the relevant mentions in the history or omits previously conveyed information assuming it is known to the system.

Therefore, in the context-dependent scenario, the contextual history is crucial to understand the follow-up questions from users, and the system often needs to reproduce partial sequences generated in previous turns. Recently, suhr2018learning proposes a context-dependent text-to-SQL model including an

interaction-level encoder and an attention mechanism over previous utterances. To reuse what has been generated, they propose to copy complete segments from the previous query. While their model is successful to reason about explicit and implicit references, it does not need to explicitly address different database schemas because the ATIS contains only the flight-booking domain. Furthermore, the model is confined to copy whole segments which are extracted by a rule-based procedure, limiting its capacity to utilize the previous query when only one or a few tokens are changed in the segment.

To exploit the correlation between sequentially generated queries and generalize the system to different domains, in this paper, we study an editing-based approach for cross-domain context-dependent text-to-SQL generation task. We propose query generation by editing the query in the previous turn. To this end, we first encode the previous query as a sequence of tokens, and the decoder computes a switch to change it at the token level. This sequence editing mechanism models token-level changes and is thus robust to error propagation. Furthermore, to capture the user utterance and the complex database schemas in different domains, we use an utterance-table encoder based on BERT to jointly encode the user utterance and column headers with co-attention, and adopt a table-aware decoder to perform SQL generation with attentions over both the user utterance and column headers.

We evaluate our model on SParC BIBREF0, a new large-scale dataset for cross-domain semantic parsing in context consisting of coherent question sequences annotated with SQL queries over 200 databases in 138 domains. Experiment results show that by generating from the previous query, our model delivers an improvement of 7% question match accuracy and 11% interaction match accuracy over the previous state-of-the-art. Further analysis shows that our editing approach is more robust to error propagation than copying segments, and the improvement becomes more significant if the basic text-to-SQL generation accuracy (without editing) improves.

Cross-Domain Context-Depencent Semantic Parsing ::: Datasets

We use SParC BIBREF0, a large-scale cross-domain context-dependent semantic parsing dataset with SQL labels, as our main evaluation benchmark. A SParC example is shown in Table TABREF1. We also report performance on ATIS BIBREF1, BIBREF2 for direct comparison to suhr2018learning. In addition, we evaluate the cross-domain context-independent text-to-SQL ability of our model on Spider BIBREF3, which SParC is built on.

We summarize and compare the data statistics in Table and Table . While the ATIS dataset has been extensively studied, it is limited to a particular domain. By contrast, SParC is both context-dependent and cross-domain. Each interaction in SParC is constructed using a question in Spider as the interaction goal, where the annotator asks inter-related questions to obtain information that completes the goal. SParC contains interactions over 200 databases and it follows the same database split as Spider where each database appears only in one of train, dev and test sets. In summary, SParC introduces new challenges to context-dependent text-to-SQL because it (1) contains more complex context dependencies, (2) has greater semantic coverage, and (3) adopts a cross-domain task setting.

Cross-Domain Context-Depencent Semantic Parsing ::: Task Formulation

Let X denote a natural language utterance and Y denote the corresponding SQL query. Context-independent semantic parsing considers individual (X,Y) pairs and maps X to Y. In context-dependent semantic parsing, we consider an interaction I consisting of n utterance-query pairs in a sequence:

At each turn t, the goal is to generate Y_t given the current utterance X_t and the interaction history

Furthermore, in the cross-domain setting, each interaction is grounded to a different database. Therefore,

the model is also given the schema of the current database as an input. We consider relational databases with multiple tables, and each table contains multiple column headers:

where m is the number of column headers, and each c_l consists of multiple words including its table name and column name (§ SECREF11).

Methodology

We employ an encoder-decoder architecture with attention mechanisms BIBREF4, BIBREF5 as illustrated in Figure FIGREF2. The framework consists of (1) an utterance-table encoder to explicitly encode the user utterance and table schema at each turn, (2) A turn attention incorporating the recent history for decoding, (3) a table-aware decoder taking into account the context of the utterance, the table schema, and the previously generated query to make editing decisions.

Methodology ::: Utterance-Table Encoder

An effective encoder captures the meaning of user utterances, the structure of table schema, and the relationship between the two. To this end, we build an utterance-table encoder with co-attention between the two as illustrated in Figure FIGREF7.

Figure FIGREF7 shows the utterance encoder. For the user utterance at each turn, we first use a bi-LSTM to encode utterance tokens. The bi-LSTM hidden state is fed into a dot-product attention layer BIBREF5 over the column header embeddings. For each utterance token embedding, we get an attention weighted average of the column header embeddings to obtain the most relevant columns BIBREF6. We then concatenate the bi-LSTM hidden state and the column attention vector, and use a second layer bi-LSTM to generate the utterance token embedding $\mathbf {h}^{E}$.

Figure FIGREF7 shows the table encoder. For each column header, we concatenate its table name and its column name separated by a special dot token (i.e., table name . column name). Each column header is processed by a bi-LSTM layer. To better capture the internal structure of the table schemas (e.g., foreign key), we then employ a self-attention BIBREF7 among all column headers. We then use an attention layer to capture the relationship between the utterance and the table schema. We concatenate the self-attention vector and the utterance attention vector, and use a second layer bi-LSTM to generate the column header embedding $\mathbf {h}^{C}$.

Note that the two embeddings depend on each other due to the co-attention, and thus the column header representation changes across different utterances in a single interaction.

Utterance-Table BERT Embedding. We consider two options as the input to the first layer bi-LSTM. The first choice is the pretrained word embedding. Second, we also consider the contextualized word embedding based on BERT BIBREF8. To be specific, we follow hwang2019comprehensive to concatenate the user utterance and all the column headers in a single sequence separated by the [SEP] token:

This sequence is fed into the pretrained BERT model whose hidden states at the last layer is used as the input embedding.

Methodology ::: Interaction Encoder with Turn Attention

To capture the information across different utterances, we use an interaction-level encoder BIBREF9 on top of the utterance-level encoder. At each turn, we use the hidden state at the last time step from the utterance-level encoder as the utterance encoding. This is the input to a uni-directional LSTM interaction encoder:

The hidden state of this interaction encoder $\mathbf {h}^{I}$ encodes the history as the interaction proceeds.

Turn Attention When issuing the current utterance, the user may omit or explicitly refer to the previously mentioned information. To this end, we adopt the turn attention mechanism to capture correlation between the current utterance and the utterance(s) at specific turn(s). At the current turn t, we compute the turn attention by the dot-product attention between the current utterance and previous utterances in the history, and then add the weighted average of previous utterance embeddings to the current utterance embedding:

The $\mathbf {c}_{t}^{\text{turn}}$ summarizes the context information and the current user query and will be used as the initial decoder state as described in the following.

Methodology ::: Table-aware Decoder

We use an LSTM decoder with attention to generate SQL queries by incorporating the interaction history, the current user utterance, and the table schema.

Denote the decoding step as k, we provide the decoder input as a concatenation of the embedding of SQL query token $\mathbf {q}_k$ and a context vector $\mathbf {c}_k$:

where $\mathbf {h}^{D}$ is the hidden state of the decoder LSTM^{D}, and the hidden state $\mathbf {h}^{D}_{0}$ is initialized by $\mathbf {c}_{t}^{\text{turn}}$. When the query token is a SQL keyword, $\mathbf {q}_k$ is a learned embedding; when it is a column header, we use the column header embedding given by the table-utterance encoder as $\mathbf {q}_k$. The context vector $\mathbf {c}_k$ is described below.

Context Vector with the Table and User Utterance. The context vector consists of attentions to both the table and the user utterance. First, at each step k, the decoder computes the attention between the decoder hidden state and the column header embedding.

where l is the index of column headers and $\mathbf {h}^{C}_{l}$ is its embedding. Second, it also computes the attention between the decoder hidden state and the utterance token embeddings:

where i is the turn index, j is the token index, and $\mathbf {h}^{E}_{i,j}$ is the token embedding for the j-th token of i-th utterance. The context vector $\mathbf {c}_k$ is a concatenation of the two:

Output Distribution. In the output layer, our decoder chooses to generate a SQL keyword (e.g., SELECT, WHERE, GROUP BY, ORDER BY) or a column header. This is critical for the cross-domain setting where the table schema changes across different examples. To achieve this, we use separate layers to score SQL keywords and column headers, and finally use the softmax operation to generate the output probability distribution:

Methodology ::: Query Editing Mechanism

In an interaction with the system, the user often asks a sequence of closely related questions to complete the final query goal. Therefore, the query generated for the current turn often overlaps significantly with the previous ones.

To empirically verify the usefulness of leveraging the previous query, we consider the process of generating the current query by applying copy and insert operations to the previous query. Figure

FIGREF18 shows the SQL query length and the number of copy and insert operations at different turns. As the interaction proceeds, the user question becomes more complicated as it requires longer SQL query to answer. However, more query tokens overlap with the previous query, and thus the number of new tokens remains small at the third turn and beyond.

Based on this observation, we extend our table-ware decoder with a query editing mechanism. We first encode the previous query using another bi-LSTM, and its hidden states are the query token embeddings $\mathbf {h}^{Q}_{i,j^{\prime }}}$ (i.e., the $j^{\prime }$-th token of the i-th query). We then extend the context vector with the attention to the previous query:

where $\mathbf {c}_k^{\text{query}}}$ is produced by an attention to query tokens $\mathbf {h}^{Q}_{i,j^{\prime }}}$ in the same form as Equation DISPLAY_FORM16.

At each decoding step, we predict a switch $p_{\text{copy}}}$ to decide if we need copy from the previous query or insert a new token.

Then, we use a separate layer to score the query tokens at turn $t-1$, and the output distribution is modified as the following to take into account the editing probability:

While the copy mechanism has been introduced by gu2016incorporating and see2017get, they focus on summarization or response generation applications by copying from the source sentences. By contrast, our focus is on editing the previously generated query while incorporating the context of user utterances and table schemas.

Related Work

Semantic parsing is the task of mapping natural language sentences into formal representations. It has been studied for decades including using linguistically-motivated compositional representations, such as logical forms BIBREF10, BIBREF11 and lambda calculus BIBREF12, BIBREF13, and using executable programs, such as SQL queries BIBREF14, BIBREF15 and other general-purpose programming languages BIBREF16, BIBREF17. Most of the early studies worked on a few domains and small datasets such as GeoQuery BIBREF10 and Overnight BIBREF18.

Recently, large and cross-domain text-to-SQL datasets such as WikiSQL BIBREF15 and Spider BIBREF3 have received an increasing amount of attention as many data-driven neural approaches achieve promising results BIBREF19, BIBREF20, BIBREF21, BIBREF22, BIBREF23, BIBREF24, BIBREF25, BIBREF6, BIBREF26, BIBREF27, BIBREF28, BIBREF29, BIBREF30. Most of them still focus on context-independent semantic parsing by converting single-turn questions into executable queries.

Relatively less effort has been devoted to context-dependent semantic parsing on datasets including ATIS BIBREF1, BIBREF31, SpaceBook BIBREF32, SCONE BIBREF33, BIBREF34, BIBREF35, BIBREF36, BIBREF37, SequentialQA BIBREF38, SParC BIBREF0 and CoSQL BIBREF39. On ATIS, miller1996fully maps utterances to semantic frames which are then mapped to SQL queries; zettlemoyer2009learning starts with context-independent Combinatory Categorial Grammar (CCG) parsing and then resolves references to generate lambda-calculus logical forms for sequences of sentences. The most relevant to our work is suhr2018learning, who generate ATIS SQL queries from interactions by incorporating history with an interaction-level encoder and copying segments of previously generated queries. Furthermore, SCONE contains three domains using stack- or list-like elements and most queries include a single binary predicate. SequentialQA is created by decomposing some complicated questions in WikiTableQuestions BIBREF40. Since both SCONE and SequentialQA are

annotated with only denotations but not query labels, they don't include many questions with rich semantic and contextual types. For example, SequentialQA BIBREF38 requires that the answer to follow-up questions must be a subset of previous answers, and most of the questions can be answered by simple SQL queries with SELECT and WHERE clauses.

Concurrent with our work, yu2019cosql introduced CoSQL, a large-scale cross-domain conversational text-to-SQL corpus collected under the Wizard-of-Oz setting. Each dialogue in CoSQL simulates a DB querying scenario with a crowd worker as a user and a college computer science student who is familiar with SQL as an expert. Question-SQL pairs in CoSQL reflect greater diversity in user backgrounds compared to other corpora and involve frequent changes in user intent between pairs or ambiguous questions that require user clarification. These features pose new challenges for text-to-SQL systems.

Our work is also related to recently proposed approaches to code generation by editing BIBREF41, BIBREF42, BIBREF43. While they follow the framework of generating code by editing the relevant examples retrieved from training data, we focus on a context-dependent setting where we generate queries from the previous query predicted by the system itself.

Experimental Results

Experimental Results ::: Metrics

On both Spider and SParC, we use the exact set match accuracy between the gold and the predicted queries . To avoid ordering issues, instead of using simple string matching, yu2018spider decompose predicted queries into different SQL clauses such as SELECT, WHERE, GROUP BY, and ORDER BY

and compute scores for each clause using set matching separately. On SparC, we report two metrics: question match accuracy which is the score average over all questions and interaction match accuracy which is average over all interactions.

Experimental Results ::: Baselines

SParC. We compare with the two baseline models released by yu2019sparc. (1) Context-dependent Seq2Seq (CD-Seq2Seq): This model is adapted from suhr2018learning. The original model was developed for ATIS and does not take the database schema as input hence cannot generalize well across domains. yu2019sparc adapt it to perform context-dependent SQL generation in multiple domains by adding a bi-LSTM database schema encoder which takes bag-of-words representations of column headers as input. They also modify the decoder to select between a SQL keyword or a column header.

(2) SyntaxSQL-con: This is adapted from the original context-agnostic SyntaxSQLNet BIBREF44 by using bi-LSTMs to encode the interaction history including the utterance and the associated SQL query response. It also employs a column attention mechanism to compute representations of the previous question and SQL query.

Spider. We compare with the results as reported in yu2018syntaxsqlnet. Furthermore, we also include recent results from lee2019recursive who propose to use recursive decoding procedure, bogin2019representing introducing graph neural networks for encoding schemas, and guo2019towards who achieve state-of-the-art performance by using an intermediate representation to bridge natural

language questions and SQL queries.

Experimental Results ::: Implementation Details

Our model is implemented in PyTorch BIBREF45. We use pretrained 300-dimensional GloVe BIBREF46 word embedding. All LSTM layers have 300 hidden size, and we use 1 layer for encoder LSTMs, and 2 layers for decoder LSTMs. We use the ADAM optimizer BIBREF47 to minimize the token-level cross-entropy loss with a batch size of 16. Model parameters are randomly initialized from a uniform distribution $U[-0.1,0.1]$. The main model has an initial learning rate of 0.001 and it will be multiplied by 0.8 if the validation loss increases compared with the previous epoch. When using BERT instead of GloVe, we use the pretrained small uncased BERT model with 768 hidden size, and we fine tune it with a separate constant learning rate of 0.00001. The training typically converges in 10 epochs.

Experimental Results ::: Overall Results

Spider. Table TABREF28 shows the results on Spider dataset. Since each question is standalone, we don't use interaction-level decoder or query editing. Our method can achieve the performance of 36.4% on dev set and 32.9% on test set, serving as a strong model for the context-independent cross-domain text-to-SQL generation. This demonstrates the effectiveness of our utterance-table encoder and table-aware decoder to handle the semantics of user utterances and the complexity of table schemas to generate complex SQL queries in unseen domains.

Furthermore, adding the utterance-table BERT embedding gives significant improvement, achieving 57.6% on dev set and 53.4% on test set, which is comparable to the state-of-the-art results from IRNet with BERT. We attribute our BERT model's high performance to (1) the empirically powerful text understanding ability of pretrained BERT model and (2) the early interaction between utterances and

column headers when they are concatenated in a single sequence as the BERT input.

SParC. Table shows the results on SParC dataset. Similar to Spider, our model without previous query as input already outperforms SyntaxSQL-con, achieving 31.4% question matching accuracy and 14.7% interaction matching accuracy. In addition, compared with CD-Seq2Seq, our model enjoys the benefits of the table-utterance encoder, turn attention, and the joint consideration of utterances and table schemas during the decoding stage. This boosts the performance by 10% question accuracy and 6% interaction accuracy.

Furthermore, we also investigate the effect of copying segment. We use the same segment copy procedure as suhr2018learning: first deterministically extract segments from the previous query and encode each segment using an LSTM, then generate a segment by computing its output probability based on its segment encoding. However, since the segment extraction from suhr2018learning is exclusively designed for the ATIS dataset, we implement our own segment extraction procedure by extracting SELECT, FROM, GROUP BY, ORDER BY clauses as well as different conditions in WHERE clauses. In this way, 3.9 segments can be extracted per SQL on average. We found that adding segment copying to CD-Seq2Seq gives a slightly lower performance on question matching and a small gain on interaction matching, while using segments extracted from the gold query can have much higher results. This demonstrates that segment copy is vulnerable to error propagation. In addition, it can only copy whole segments hence has difficulty capturing the changes of only one or a few tokens in the query.

To better understand how models perform as the interaction proceeds, Figure FIGREF30 (Left) shows the performance split by turns on the dev set. The questions asked in later turns are more difficult to answer given longer context history. While the baselines have lower performance as the turn number increases, our model still maintains 38%-48% accuracy for turn 2 and 3, and 20% at turn 4 or beyond. Similarly, Figure FIGREF30 (Right) shows the performance split by hardness levels with the frequency of

examples. This also demonstrates our model is more competitive in answering hard and extra hard questions.

ATIS. We also report our model performance on ATIS in Table . Our model achieves 36.2% dev and 43.9% test string accuracy, comparable to suhr2018learning. On ATIS, we only apply our editing mechanism and reuse their utterance encoder instead of the BERT utterance-table encoder, because ATIS is single domain.

Experimental Results ::: Effect of Query Editing

We further investigate the effect of our query editing mechanism. To this end, we apply editing from both the gold query and the predicted query on our model with or without the utterance-table BERT embedding. We also perform an ablation study to validate the contribution of query attention and sequence editing separately.

As shown in Table , editing the gold query consistently improves both question match and interaction match accuracy. This shows the editing approach is indeed helpful to improve the generation quality when the previous query is the oracle.

Using the predicted query is a more realistic setting, and in this case, the model is affected by error propagation due to the incorrect queries produced by itself. For the model without the utterance-table BERT embedding, using the predicted query only gives around 1.5% improvement. As shown in Figure FIGREF33, this is because the editing mechanism is more helpful for turn 4 which is a small fraction of all question examples. For the model with the utterance-table BERT embedding, the query generation accuracy at each turn is significantly improved, thus reducing the error propagation effect. In this case, the editing approach delivers consistent improvements of 7% increase on question matching accuracy

and 11% increase on interaction matching accuracy. Figure FIGREF33 also shows that query editing with BERT benefits all turns.

Finally, as an ablation study, Table also reports the result with only query attention (use predicted query) on the dev set. This improves over our vanilla BERT model without query attention and achieves 42.7% question and 21.6% interaction matching accuracy. With query editing, our best model further improves to 47.2% question and 29.5% interaction matching accuracy. This demonstrates the effectiveness of our query attention and query editing separately, both of which are essential to make use of the previous query.

Conclusions

In this paper, we propose an editing-based encoder-decoder model to address the problem of context-dependent cross-domain text-to-SQL generation. While being simple, empirical results demonstrate the benefits of our editing mechanism. The approach is more robust to error propagation than copying segments, and its performance increases when the basic text-to-SQL generation quality (without editing) is better.

Acknowledgements

We thank the anonymous reviewers for their thoughtful detailed comments.