Towards Faithful Response Generation for Chinese Table Question Answering

Anonymous ACL submission

Abstract

The response generation for TableQA aims to automatically generate a response to end-users from a SQL query and its corresponding execution result (in the form of table). It is an essential and practical task. However, there has been little work on it in recent years. We consider this may be blamed on the lack of large-scale and high-quality datasets in this area. In this paper, we present `ResponseNLG`, a large-scale and high-quality Chinese dataset for TableQA response generation, to advance the field in both academic and industrial communities. Further, to bridge the structural gap between the input SQL and table and establish better semantic alignments, we propose a Heterogeneous Graph Transformation approach. In this way, we establish a joint encoding space for the two heterogeneous input data and convert this task to a Graph-to-Text problem. We further introduce the Node Segment Embedding to better preserve the original graph structure upon PLMs based models.

1 Introduction

Table Question Answering (TableQA) aims to answer a question over the given tables, and it has been widely applied in many real-life applications, e.g., chatbot and business intelligence (Stent et al., 1999; Litman and Silliman, 2004; Budzianowski et al., 2018). One common solution is converting it to a Text-to-SQL problem (Warren and Pereira, 1982; Zettlemoyer and Collins, 2005; Mrksic et al., 2015), which maps the natural language question to meaning representations in SQL. Once a natural language question has been mapped to a formal SQL query, the result can be retrieved from the table database based on it. In a real-world setting, the consequent problem is how to convert the execution result, which usually can be organized as a table, to a natural language text to the asker, i.e., the response generation for TableQA (Yu et al., 2019a).

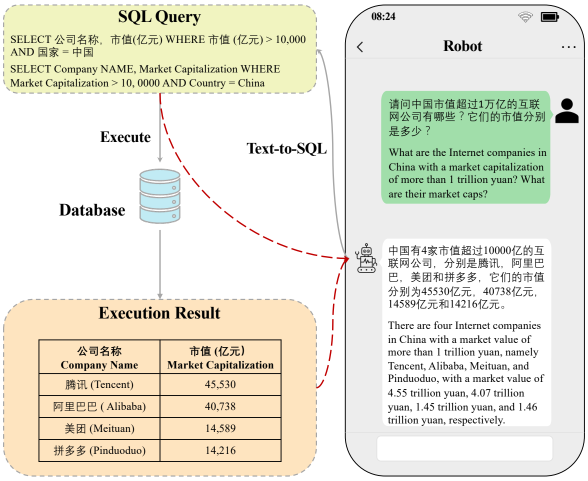


Figure 1: An example for Text-to-SQL and TableQA Response Generation. The red dotted lines denote the input data (a SQL query and its execution result) for TableQA Response Generation.

The response generation for TableQA takes a SQL query and its corresponding execution result (in the form of a table) as input and aims to generate a natural language description as the response (as shown in Figure 1). Intuitively, it plays a vital and indispensable role in constructing a real-life TableQA application and building a human-like dialog system. Meanwhile, this task is challenging. The first challenge is that the model needs to understand the two heterogeneous input data: SQL and table. Moreover, both input data are structured and have less semantic information than natural language sentences, which also exists in Table-to-Text generation (Lebret et al., 2016; Wiseman et al., 2017) but is more challenging. Additionally, the generated response must be absolutely faithful to the input data, which means the response should contain all the content in the input table while being logically consistent with the SQL.

To our knowledge, template-based models are widely applied in dialogue response generation modules (Jordan et al., 2006; Ultes et al., 2017).

The experts, who have abundant linguistic and domain knowledge, write different kinds of templates with slots which are then filled with the execution results(Ritter et al., 2011; Kale and Rastogi, 2020). Obviously, to cover more data from different domains, this system needs numerous templates, which typically require a lot of human effort and costs. Meanwhile, it is not easy to guarantee the fluency of the generated results. Over the past several years, automatic neural network-based methods have achieved significant progress in the text generation domain (Liu et al., 2016; Lubis et al., 2018). However, we notice that there is little work on response generation for TableQA. We consider this may be blamed on the lack of large-scale and high-quality datasets in this field. CoSQL (Yu et al., 2019a) is the only dataset for this task with 7,845 generation examples, and it is in English. It is a dataset with SQL-grounded dialogue state tracking as the core, and the generation annotations are very rough.

In this paper, we propose `ResponseNLG`, a large-scale and high-quality Chinese dataset for TableQA Response Generation. We introduce a dataset construction process where annotators only need to directly revise the provided template response, and yield `29,366` response generation examples. It is an order of magnitude larger than CoSQL. A strict screening procedure is implemented to ensure data quality. `ResponseNLG` has a wider distribution than CoSQL, which is more in line with real TableQA scenarios. Meanwhile, to bridge the structural gap between the input SQL and table and establish better semantic alignments, we propose a Heterogeneous Graph Transformation approach (HGT). HGT first converts the two sources to two undirected graphs and then builds the connection between the nodes in different graphs to obtain a heterogeneous joint graph. In this way, we convert this task to a Graph-to-Text problem. Previous Graph-to-Text methods (Ribeiro et al., 2020, 2021) transform the input graph into a new token graph to introduce pretrained language models (PLMs). We consider that this transformation breaks the input graph structure and may bring in extra noises into graph encoding. To preserve original structure information, we introduce the Node Segment Embedding, which assigns the same symbol to the nodes in the token graph which belong to the same node in the original heterogeneous graph. Our contributions include the following three aspects:

- We present a large-scale and high-quality Chinese dataset for TableQA response generation, `ResponseNLG`, with a series of strong baselines and metrics. To the best of our knowledge, it is also the first Chinese dataset for this task.
- We propose a Heterogeneous Graph Transformation method to bridge the structural gap between the SQL and table. We also introduce Node Segment Embedding to better preserve the original graph structure upon PLMs based models.
- Experiments and analysis on `ResponseNLG` attest to both the high quality and challenges of the dataset. The results also demonstrate the effectiveness of our proposed method. We will make our data and code publicly available upon the acceptance of this paper.

2 Related Works

2.1 Table Question Answering

A TableQA system comprises a table semantic parsing (Text-to-SQL) component and a response generation component (Yu et al., 2019a). The semantic parsing component converts NL question into SQL query (Text-to-SQL) (Finegan-Dollak et al., 2018; Guo et al., 2019; Wang et al., 2020a; Hui et al., 2021) and the response generation component generate NL response given the SQL query and SQL execution table. Notice that the SQL query can represent the context state in multi turn TableQA scenarios (Yu et al., 2019a,b). Several datasets have been proposed to apply semantic parsing on tables, including WikiTableQuestions (Pasupat and Liang, 2015), SequentialQA (Iyyer et al., 2017), WikiSQL (Zhong et al., 2017), Spider (Yu et al., 2018), SparC (Yu et al., 2019b) and CHASE (Guo et al., 2021). But these works only focus on the semantic parsing task and return the SQL execution result as simple short form answer. FeTaQA (Nan et al., 2021) yields a more challenging TableQA setting because it requires generating free-form text answers. HybridQA (Chen et al., 2020b) and OTT-QA (Chen et al., 2020a) build question answering tasks with context of both structured tables and unstructured text.

2.2 Data-to-Text Generation

Data-to-Text aims to generate a natural language description from structural or semi-structural data

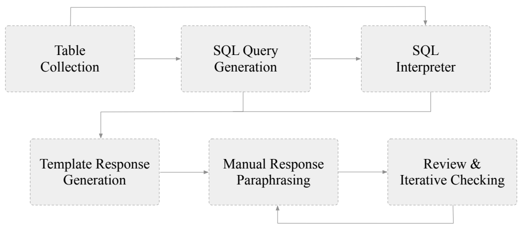


Figure 2: Construction workflow of ResponseNLG.

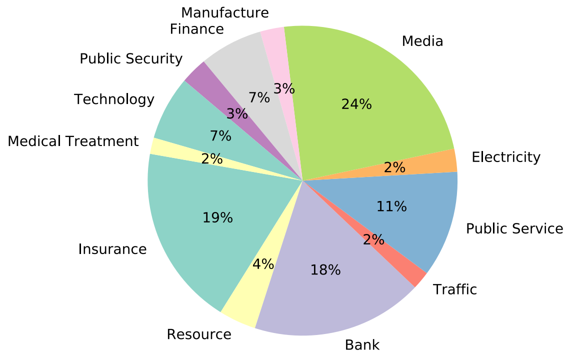
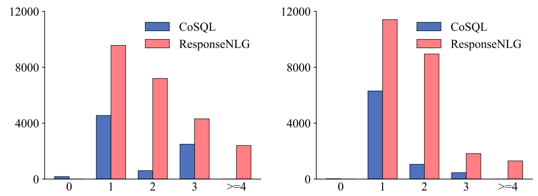


Figure 3: Topic distribution of ResponseNLG.



(a) Row Complexity (b) Column Complexity

Figure 4: The comparison of data complexity distribution between CoSQL and ResponseNLG.

(Liang et al., 2009; Banik et al., 2012; Gardent et al., 2017; Parikh et al., 2020). It helps people get the key points of the input data and makes the stored information accessible to a broader audience of end-users. In the academic community, Data-to-Text is usually divided into Graph-to-Text (Song et al., 2018; Wang et al., 2020c) and Table-to-Text (Lebret et al., 2016; Wiseman et al., 2017), according to whether input data is a graph (e.g., Knowledge or Abstract Meaning Representation Graph) or table (Ribeiro et al., 2019; Agarwal et al., 2021). To better model the structure of a graph, early works (Song et al., 2018; Koncel-Kedziorski et al., 2019; Damonte and Cohen, 2019) introduce Graph Neural Networks (GNNs) as the structure encoder, which only considers the relations between neighbor nodes. Unlike the local encoding strategies, Zhu et al.; Cai and Lam propose the Graph Transformer that uses explicit relation encoding and allows direct communication between two distant nodes. In order to learn better contextualized node embeddings, Ribeiro et al. gather the above two encoding strategies, proposing novel neural models which encode an input graph combining both global and local node contexts. To better leverage the structure of tables, some studies (Bao et al., 2018; Nema et al., 2018; Jain et al., 2018; Liu et al., 2019; Li et al., 2021) propose to utilize the hierarchal encoder to model the table's representation from the row and column levels.

The response generation for TableQA can also be regarded as a Data-to-Text task, and it is similar to Table-to-Text but more challenging because its input data contains not only a structural table but also a SQL, which are both essential for the generation. Moreover, it requires the model to generate an utterly faithful response to the input data, which means the response should contain all the content in the table and be logically consistent with the SQL.

3 Dataset Construction

3.1 Data Synthetic and Annotation

Different from previous works (Yu et al., 2019a), which usually rely on humans to create natural Language (NL) questions, SQL queries and corresponding response, we introduce a dataset construction process where annotators only need to directly revise the provided template response as illustrated in Figure 2. We first collect tables from the Internet and utilize production rules to generate SQL queries automatically. And then, we execute the SQL on the collected tables. After that, we generate a pseudo response based on predefined templates. Finally, pseudo responses are paraphrased to NL responses by humans. Additionally, to guarantee data quality, low-confidence instances are detected according to their overlapping and similarity metrics and are further checked by humans.

Table Collection We build a search engine based table collection pipeline to collect high quality tables. Firstly, 100,000 frequently used words are summarized from the CLUE (Xu et al., 2020) corpus. Then these words are queried in Google and filtered spreadsheet files are downloaded. Useful tables are extracted from these files through a parser, which could identify potential table in a worksheet. Sensitive values in the tables, such as

Passwords, Identification IDs and Credit Card IDs are replaced with special tokens. We also build a table cleaning pipeline as shown in Appendix A.1 to guarantee table quality.

Pseudo Data Generation Data syntactic for semantic parsing has gained increasing attention in recent years (Zhong et al., 2017; Wang et al., 2020b, 2021). Differently, we apply syntactic method to build response generation dataset. We firstly utilize production rules from the SQL grammar to automatically generate SQL queries. The SQL query can be represented as an abstract syntax trees (ASTs) using the rules such as `SQLs = SQL, SQL = Select Where, Select = SELECT A, Where = WHERE Conditions...`, all of which are production rules of the SQL grammar. Please refer to Appendix A.2 for more details. By exploiting every rule of the grammar, we can generate SQL queries covering patterns of different complexity along with corresponding tables. SQL querys which cannot execute or have not execution results are filtered. We then build two template-based generation pipelines. The one is to convert the syntactic SQL query into pseudo NL question. The other is to generate template NL response based on SQL query and the SQL execution result table.

Data Annotation and Review We employ 20 well-educated crowd workers to paraphrase the template questions and template response into natural language, and filter incomprehensible ones which are semantically unclear. To guarantee data quality, another 4 workers are asked to review the annotated data. Data with poor annotation quality will be required to be relabeled. We also automatically detect low-quality data. If the response does not contain important information about SQL and Table, we will filter it out.

3.2 Dataset Statistics

Our final `ResponseNLG` dataset contains `29,358` examples, with a average length of `46.7`. Each example contains a {NL question, SQL query, SQL execution table, NL response} pair. We split the training/development/test set by `23,488/2,935/2,935` randomly.

Topics We build a topic categorization model (Asthana and Halfaker, 2018) for tables in `ResponseNLG` to investigate the topics distribution. Figure 3 presents an aggregated topic analysis of our dataset. We find that the `Media`, `Insurance` and `Bank` topics together comprise 61% of our dataset, but the other 39% is composed of broader topics such as `Public Service`, `Technology`, `Finance`. Our dataset is limited to topics that are present in CLUE.

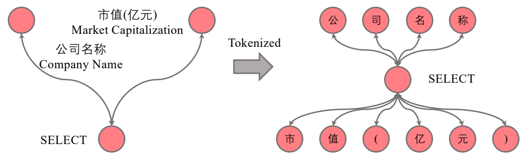
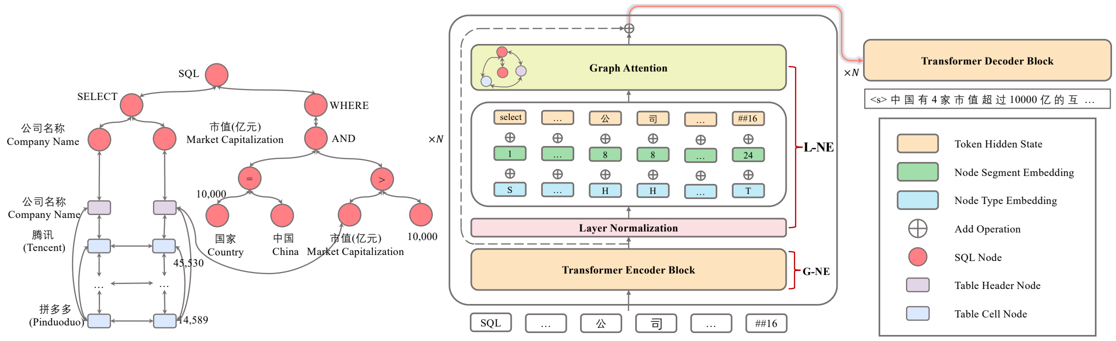


Figure 5: An example of token graph transformation.

Data Complexity We evaluate the data complexity by the row number and column number of the input tables. Figure 4 shows the training set distribution comparison between CoSQL and `ResponseNLG`. We can see that the `ResponseNLG` has a wider distribution than CoSQL, which is more in line with real TableQA scenarios. Please refer to Appendix A.3 for more details.

4 Structure-Aware Approach

Given an input SQL s and a Table t, the model aims to generate a response $\tilde{y}$. To bridge the gap between the two sources of information, we first propose a **Heterogeneous Graph Transformation** approach (HGT), which explicitly connects the input SQL and table in a heterogeneous graph structure. In this way, we can obtain a joint graph representation of the two sources and convert the response generation task to a Graph-to-Text problem. And then, we utilize a varietal transformer architecture (Ribeiro et al., 2020) that employs the original transformer encoder as the Global Node Encoder (G-NE) and introduces a GNN based layer into each transformer encoder layer as the Local Node Encoder (L-NE). G-NE allows explicit communication between two distant nodes, taking advantage of a large node context range. And L-NE has an advantage in modeling the graph topology. As shown in Figure 6 (b), this architecture cascaded performs global and local node aggregation, which gathers the benefits from both strategies. In the rest of this section, we will describe the proposed Heterogeneous Graph Transformation approach and the Local Node Encoder in detail.



(a) Heterogeneous Graph

(b) Overview of Architecture

Figure 6: Illustration of the proposed method. (a) is an example of a heterogeneous graph transformed by Heterogeneous Graph Transformation. (b) is an overview of our model. **L-NE** and **G-NE** denote Local Node Encoder and Global Node Encoder, respectively.

4.1 Heterogeneous Graph Transformation

Given a SQL s and its execution result (in the form of a table) t as input (shown in Figure 1), the Heterogeneous Graph Transformation approach takes two steps to transform the input two sources of data into a heterogeneous graph (shown in Figure 6a). First, it converts the SQL and table into two undirected graphs: SQL graph $\mathcal{G}_s$ and table graph $\mathcal{G}_t$. In particular, for a SQL, we follow the previous method (Xu et al., 2018) and convert it to a tree. We refer the readers to the paper for more details. For a table, we treat each column name and table cell as a node and divide the nodes in the table into two categories: table header node and table cell node. And then, we connect each header node with the cell node in the same column. We also build the connections between the cell nodes in the same row. Second, we add connections between the nodes that indicate the same column in $\mathcal{G}_s$ and $\mathcal{G}_t$ to build the unified heterogeneous graph. we also add a self-loop connection for each node. The transformed heterogeneous graph is formulated as $\mathcal{G}_h = (\mathcal{V}_h, \mathcal{E}_h)$, where $\mathcal{V}$ represents the nodes set and $\mathcal{E}_h = \{(n, v)|n, v \in \mathcal{V}\}$. Figure 6a shows an example of the transformed heterogeneous graph.

We expect that developing generation model should benefit from the recent advance on pre-trained language models (PLMs) (Peters et al., 2018; Devlin et al., 2019; Radford et al., 2019). We represent each $\mathcal{G}_h$ using subword tokens, and convert it into a new token graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Specifically, each token of a node in V_h becomes a node $\tilde{v}$ in $\mathcal{N}$. For each edge $(n, v) \in \mathcal{E}_h$, we connect each token between n and v to obtain the new edges set $\mathcal{E}$ (as shown in Figure 5). However, we notice that the new token graph $\mathcal{G}$ breaks the structure of the original graph $\mathcal{G}_h$ and may make the encoder pay too much attention to the feature of nodes at the token level instead of the original node level. This may bring extra noises into graph encoding. To preserve the original structural information, we introduce the **Node Segment Embedding** (NSE), which assigns the same symbol to the nodes in the token graph $\mathcal{G}$ which belong to the same node in the original heterogeneous graph $\mathcal{G}_h$.

4.2 Local Node Encoder

Given $\{h_v|v \in \mathcal{V}\}$ as the outputs of the Global Node Encoder at the L-th encoder layer, we next describe how the Local Node Encoder works. As shown in Figure 6b, the Local Node Encoder consists of two main modules: Node Embedding Layer and Graph Attention Network Layer. The former enriches the features of the nodes, and the latter explicitly models the graph structure. For Node Embedding Layer, in addition to the above Node Segment Embedding, we also introduce Node Type Embedding (NTE) to preserve the graph heterogeneity. Formally, given h_v, we obtain the feature-enhanced node representation by:

$$h_v^e = LayerNorm(h_v) + e_v^s + e_v^t \qquad (1)$$

where $LayerNorm$ represents layer normalization (Ba et al., 2016). e_v^s, e_v^t denote the node segment embedding and node type embedding for node v respectively.

After the Node Embedding Layer, we utilize Graph Neural Networks (GNNs) to model the graph structure explicitly. For simplicity, we employ one Graph Attention Network Layer (GAT). Formally, it aggregates the representations of node v in a multi-head self-attention layer (Vaswani

Model	BLEU	BLEU-2	BLEU-4	CHRF++	PARENT-P	PARENT-R	PARENT
				Development			
Pointer-Generator	50.30	54.40	37.30	59.46	53.22	80.95	62.91
Finetune	53.62	58.88	39.60	62.67	56.58	84.35	66.63
Finetune-FNN	54.32	59.53	40.25	63.06	56.67	84.49	66.73
Finetune-Graph	52.42	57.85	38.30	60.78	56.57	83.76	66.42
Finetune-Graph-FNN	53.03	58.45	38.63	60.97	**56.79**	83.98	66.64
Ours	**55.88***	**60.88***	**42.15***	**63.94***	56.75	**84.81**	**66.91**
				Test			
Pointer-Generator	50.12	55.35	37.15	58.67	54.13	80.36	63.22
Finetune	53.78	58.98	39.78	62.81	56.63	84.25	66.65
Finetune-FNN	54.18	59.45	40.08	62.95	**56.82**	84.41	**66.85**
Finetune-Graph	52.77	58.18	38.43	61.07	56.52	83.63	66.37
Finetune-Graph-FNN	53.34	58.75	39.00	61.41	56.63	84.08	66.60
Ours	**55.80***	**60.78***	**42.08***	**64.02***	56.58	**84.63**	66.73

Table 1: Main results of models on `ResponseNLG` development set. * denotes the value is significantly different from other models at a $p < 0.05$ level, according to an independent sample t-test.

et al., 2017) as follows:

$$\begin{aligned} s^h_{v,n} &= \frac{h^e_v W^h_Q (h^e_n W^h_K)^\top}{\sqrt{d/H}} \\ \alpha^h_{v,n} &= \frac{e^{s^h_{v,n}}}{\sum_{\tilde{n}\in\mathcal{N}(v)} e^{s^h_{v,\tilde{n}}}} \\ z^h &= \sum_{n\in\mathcal{N}(v)} \alpha^h_{v,n}(h^e_n W^h_V) \\ h^r &= Concat(z^1, ..., z^H) \end{aligned} \tag{2}$$

where $1 \leq h \leq H$, and W^h_Q, W^h_K, $W^h_V \in \mathbb{R}^{d\times(d/H)}$. $\mathcal{N}(v)$ denotes the immediate neighborhood of node v in graph $\mathcal{G}$. We also tried the RGAT (Shaw et al., 2018). It performed comparable with GAT but introduced more parameters.

4.3 Training Objective

The transformer parameters are initialized with the pretrained T5 (Raffel et al., 2020), and the others are randomly initialized. Given each gold instance (s, t, y), we fine-tune the model to optimize the following cross-entropy objective:

$$\mathcal{L} = -\sum_{i=1}^{|y|} p_\theta(y_t|y_{1:t-1}; s, t) \tag{3}$$

5 Experiment

5.1 Experiment Settings

Baselines We conduct experiments on `ResponseNLG` and compare our method with several baselines, including:

- **Pointer-Generator** is an RNN-based Seq2Seq model with attention and copy mechanism. We concatenate the SQL and linearized table as input.
- **Finetune** denotes a Transformer encoder-decoder method which is initialized by T5. It takes the same strategy with Pointer-Generator to preprocess the input SQL and table. Moreover, we replace our local graph encoder with an FNN layer. And we change the hidden dimension of FNN and make its parameters equal with the local graph encoder to make a fair comparison. We denote this method as **Finetune-FNN**.
- **Finetune-Graph** is also a T5 initialized method. Different from Finetune, it uses the same graph linearization as input with our method. Additionally, we add FNN to make a fair comparison, which is denoted as **Finetune-Graph-FNN**.

Evaluation Metrics We evaluate our models by applying both automatic and human evaluations. For automatic evaluation, we first employ two widely used metrics: BLEU (Papineni et al., 2002) and CHRF++ (Popović, 2015). We also report the results of BLEU-2 and BLUE-4. All above scores are calculated by SacreBLEU (Post, 2018). Then we employ PARENT (Dhingra et al., 2019) to evaluate the faithfulness for the generated text. PARENT is a metric proposed specifically for data-to-text evaluation that takes the table into account. We modify it to make it suitable for our dataset, described in Appendix A.4. We conduct experiments over 4 different seeds and report the average scores on them. Please refer to Section 5.4 for human evaluation details.

Implement Details Our implementation is based on Hugging Face Transformer models (Wolf et al., 2020). We utilize T5_{base} for all experiments.

Model	BLEU	CHRF++	PAR
Finetune-Graph-FNN	53.03	60.97	66.64
Finetune-Graph-L-NE	55.18	62.94	**67.18**
+ NTE	55.54	63.32	67.06
+ NSE	55.82	63.73	66.82
+ NTE & NSE	**55.88**	**63.94**	66.91

Table 2: Ablation study on `ResponseNLG` development set. PAR denotes PARENT.

Model	BLEU	CHRF++	PARENT
Finetune-FNN	54.32	64.06	66.73
-w/o SQL	41.24	46.38	50.52
-w/o TABLE	16.69	29.81	43.00
Ours	55.88	63.94	66.91
-w/o SQL	46.28	49.91	51.51
-w/o TABLE	15.53	29.20	42.98

Table 3: Effect of input SQL and Table.

For T5-based methods, we use AdamW optimizer (Loshchilov and Hutter, 2018) and employ a linearly decreasing learning rate schedule without warm-up. Moreover, the learning rate is fixed as $3e-5$, and batch size is set as 4 for all experiments. We train the parameters from T5 and the added parameters together. During decoding, we employ beam search with a beam size 5. All experiments are implemented with Pytorch and trained on Nvidia Telsa V100 32GP GPUs.

5.2 Main Result

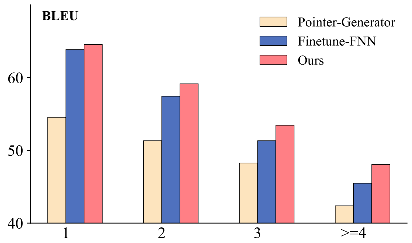
The results on `ResponseNLG` development and test sets are summarized in Table 1. First, we observe that after adding new parameters, Finetune-FNN and Finetune-Graph-FNN achieve better performance than their baselines. And then, we notice that Finetune-FNN performs better than Finetune-Graph-FNN, though their parameters are equal. We consider the reason is that the input of the former is more similar to natural language. This indicates that the representation of input data affects the model performance. Our method significantly outperforms Finetune-Graph-FNN on BLEU (**+2.85**) and CHRF++ (**+2.97**) and also obtains a higher PARENT score. It demonstrates that the improvement of our method not only comes from more parameters. Our approach also performs better than Finetune-FNN on BLEU (**+1.56**) and CHRF++ (**+0.8**) and achieves competitive results on PARENT. The results on the test set follow a pattern similar to the development set and our method achieves the start-of-the-art results on BLEU and CHRF++. It demonstrates the effectiveness of our proposed method.

5.3 Analysis and Discussion

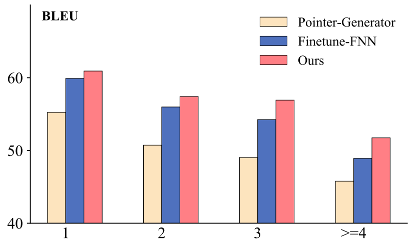
Ablation Study To examine the impact of each module in our method, we conduct the ablation study on `ResponseNLG` development set, and the results are shown in Table 2. Finetune-Graph-L-NE denotes the method that replaces each FNN module in Finetune-Graph-FNN with a GAT layer. As can be seen, the most improvement comes from the explicitly modeling of the graph structure. Moreover, both Node Type Embedding (NTE) and Node Segment Embedding (NSE) can improve the model's performance. However, they reduce the model's performance on PARENT. We think that it may be due to fluctuation of the PARENT metric.

Effects of input SQL and Table In order to examine the effects of different input data, we conduct further experiments by removing the input SQL and Table. The results are summarized in Table 3. We first remove the SQL and only utilize the Table as input. As we can see, both Finetune-FNN and our methods perform poorly on all metrics. And then, we only employ SQL as the model input. The performance degrades even more. The results demonstrate that both input SQL and table are essential for the response generation. It worth noting that Finetune-FNN and our method still obtain high PARENT scores after removing the Table input. It is unreasonable because each ground-truth response must contain all content in the input table (high coverage rate) to achieve a high faithfulness (refer to Section 5.4). Therefore, we think PARENT may not accurately measure the faithfulness of the text in `ResponseNLG`. We also notice that, after removing the input SQL, our method still performs better than Fintune-FNN. The result indicates that, in addition to using a hierarchical encoder, it may be a good choice to transform a table into a graph representation to model its structure in Table-to-Text. We leave this for future work.

Impact on the Table Complexity In order to have a deeper understanding of the model's performance, we further explore the mode performance under various numbers of rows and columns of the input table on the `ResponseNLG` development set. Figure 7 shows the BLEU comparison between our model and baselines. The BLEU scores of all the models decrease as the number of table rows or columns increases. Intuitively, the more rows or columns the table contains, the more complex the



(a) Row complexity results



(b) Column complexity results

Figure 7: Experiment results on different complexity data on `ResponseNLG` development set.

table will be. The results show that all these models are better at handling simple rather than complex tables. We observe that the improvement of our model increases as the number of rows or columns increases. In other words, our model is better at handling complex tables than other methods.

5.4 Human Evaluation

We conduct human evaluation following Parikh et al. (2020). We compare our method with Pointer-Generator, Finetune-FNN and Ortacel. Specifically, we first randomly select 100 examples from the `ResponseNLG` test set and the corresponding outputs generated by each model. And then, four annotators are asked to evaluate the quality from the following four axes:

- **Fluency**: a sentence is fluent if it is grammatical and natural. And it is scored from 1 to 10, where 1 represents not Fluent, and 10 represents Mostly Fluent.
- **Faithfulness**: a sentence is considered faithful if it is logically consistent with the input SQL and all pieces of information are supported by the table. The score ranges from 1 to 10.
- **Coverage**: percentage of cells in the input table the candidate sentence covers. It is calculated by $\frac{n^c}{n^t}$, where n^t denotes all cells in the input table, and n^c represents the number of cells covered by the sentence.

Model	Flu ↑	Fai ↑	Cov(%)↑	Rep ↓
Oracel	8.71	9.47	95.98	0.13
Pointer-Generator	6.24	6.82	85.78	0.42
Finetune-FNN	7.17	7.61	91.87	0.15
Ours	**7.39**	**7.83**	**93.15**	0.17

Table 4: Human evaluation over references (denoted as Oracle) and model outputs. Flu, Fai, Cov, Rep denote Fluency, Faithfulness, Coverage and Repetition. ↑ indicates higher is better and ↓ denotes lower is better.

- **Repetition** number of cells the candidate sentence repeats. If a cell is repeated n times, it will be recorded n times.

We also introduce the reference as one candidate. And its results can be regarded as the upper bound (denoted as Oracle). For each sample, the annotators need to evaluate four sentences based on the input data. And they do not know which model generates these sentences. The final score for each criterion is the average from all annotators.

The results summarized in Table 4 show that the **Oracle** consistently achieves high performance than generation methods. It attests to the high quality of our human annotations. Our method outperforms baselines on almost all axes. It demonstrates the effectiveness of our proposed method. Although our model achieves a high coverage rate (93.15%), its Faithfulness score is relatively low (only 7.83), and there is a considerable gap compared with the Oracle. It indicates simply copying content from the input table can not guarantee the faithfulness of the generated response. It may be necessary for the model to understand the input SQL and table deeper, which is the biggest challenge in this dataset.

6 Conclusion

We present `ResponseNLG`, a large-scale and high-quality Chinese dataset for TableQA response generation, along with a series of baselines and metrics. We build a Heterogeneous Graph Transformation method to bridge the structural gap between the SQL and table. Meanwhile, to better use PLMs, we introduce the Node Segment Embedding to solve the problem that transforming the input graph to a new token graph breaks the original graph's structure. Experiments on our `ResponseNLG` dataset show that our proposed model outperforms existing baseline models. We will make our data and code publicly available upon the acceptance of this paper.

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565.

Sumit Asthana and Aaron Halfaker. 2018. With few eyes, all hoaxes are deep. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–18.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Eva Banik, Eric Kow, Nikhil Dinesh, Vinay Chaudhri, and Umangi Oza. 2012. Natural language generation for a smart biology textbook. In *INLG 2012 Proceedings of the Seventh International Natural Language Generation Conference*, pages 125–127.

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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 *EMNLP*.

Deng Cai and Wai Lam. 2020. Graph transformer for graph-to-sequence learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7464–7471.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Yang Wang, and William W Cohen. 2020a. Open question answering over tables and text. In *International Conference on Learning Representations*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020b. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1026–1036.

Marco Damonte and Shay B Cohen. 2019. Structural neural encoders for amr-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3649–3658.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895.

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for nlg micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188.

Jiaqi Guo, Ziliang Si, Yu Wang, Qian Liu, Ming Fan, Jian-Guang Lou, Zijiang Yang, and Ting Liu. 2021. Chase: A large-scale and pragmatic chinese dataset for cross-database context-dependent text-to-sql. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2316–2331.

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 Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535.

Binyuan Hui, Ruiying Geng, Qiyu Ren, Binhua Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, Pengfei Zhu, and Xiaodan Zhu. 2021. Dynamic hybrid relation exploration network for cross-domain context-dependent semantic parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13116–13124.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831.

Parag Jain, Anirban Laha, Karthik Sankaranarayanan, Preksha Nema, Mitesh M Khapra, and Shreyas Shetty. 2018. A mixed hierarchical attention based encoder-decoder approach for standard table summarization. In *Proceedings of the 2018 Conference of* *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 622–627.

Pamela W Jordan, Maxim Makatchev, Umarani Pappuswamy, Kurt VanLehn, and Patricia L Albacete. 2006. A natural language tutorial dialogue system for physics. In *FLAIRS Conference*, pages 521–526.

Mihir Kale and Abhinav Rastogi. 2020. Template guided text generation for task-oriented dialogue. *arXiv preprint arXiv:2004.15006*.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

Liang Li, Can Ma, Yinliang Yue, and Dayong Hu. 2021. Improving encoder by auxiliary supervision tasks for table-to-text generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5979–5989.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99.

Diane Litman and Scott Silliman. 2004. Itspoke: An intelligent tutoring spoken dialogue system. In *Demonstration papers at HLT-NAACL 2004*, pages 5–8.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*.

Tianyu Liu, Fuli Luo, Qiaolin Xia, Shuming Ma, Baobao Chang, and Zhifang Sui. 2019. Hierarchical encoder with auxiliary supervision for neural table-to-text generation: Learning better representation for tables. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6786–6793.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Nurul Lubis, Sakriani Sakti, Koichiro Yoshino, and Satoshi Nakamura. 2018. Eliciting positive emotion through affect-sensitive dialogue response generation: A neural network approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Nikola Mrksic, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Pei-hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *ACL (2)*.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Nick Schoelkopf, Riley Kong, Xiangru Tang, et al. 2021. Fetaqa: Free-form table question answering. *arXiv preprint arXiv:2104.00369*.

Preksha Nema, Shreyas Shetty, Parag Jain, Anirban Laha, Karthik Sankaranarayanan, and Mitesh M Khapra. 2018. Generating descriptions from structured data using a bifocal attention mechanism and gated orthogonalization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1539–1550.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. Totto: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186.

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.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Maja Popović. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395.

Matt Post. 2018. A call for clarity in reporting bleu scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Leonardo FR Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing amr-to-text generation with dual graph representations. 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)*, pages 3183–3194.

Leonardo FR Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020. Modeling global and local node contexts for text generation from knowledge graphs. *Transactions of the Association for Computational Linguistics*, 8:589–604.

Leonardo FR. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021. Structural adapters in pretrained language models for amr-to-text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 583–593.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626.

Amanda Stent, John Dowding, Jean Mark Gawron, Elizabeth Owen Bratt, and Robert C Moore. 1999. The commandtalk spoken dialogue system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 183–190.

Stefan Ultes, Lina M Rojas Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Inigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien Wen, Milica Gasic, et al. 2017. Pydial: A multi-domain statistical dialogue system toolkit. In *Proceedings of ACL 2017, System Demonstrations*, pages 73–78.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

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*, pages 7567–7578.

Bailin Wang, Wenpeng Yin, Xi Victoria Lin, and Caiming Xiong. 2021. Learning to synthesize data for semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2760–2766.

Lijie Wang, Ao Zhang, Kun Wu, Ke Sun, Zhenghua Li, Hua Wu, Min Zhang, and Haifeng Wang. 2020b. Chitesql: A large-scale and pragmatic chinese text-to-sql dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6923–6935.

Tianming Wang, Xiaojun Wan, and Shaowei Yao. 2020c. Better amr-to-text generation with graph structure reconstruction. In *IJCAI*, pages 3919–3925.

David HD Warren and Fernando CN Pereira. 1982. An efficient easily adaptable system for interpreting natural language queries. *American journal of computational linguistics*, 8(3-4):110–122.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, et al. 2020. Clue: A chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4762–4772.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, et al. 2019a. 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)*, pages 1962–1979.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 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*, 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, et al. 2019b. Sparc: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 658–666.

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

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better amr-to-text generation. 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)*, pages 5459–5468.

A Appendix

A.1 Table Data Cleaning

We build a rule-based table cleaning pipeline to guarantee table quality. We filter out noise tables via rules as follows. There are 24K tables in our dataset.

- **Blacklist Filtering** We first builds a blacklist including special chars, dirty words, emojis, and HTML words. And filter tables if the headers or the values include any word in the blacklist.
- **Header Type Filtering** We recognize all of the header types in each table including `Text`, `Number`, `Time`, and `Bool`. If the proportion of `Text` type is less than 30%, we filter out the table.
- **Complexity Filtering** We will filter out tables with less than 2 columns or rows.
- **Repetition Filtering** If a value repeats more than 50% in a table, we will filter out the table.

A.2 SQL Query Generation

We utilize production rules from the SQL grammar to automatically generate SQL queries inspired by Zhong et al. (2017); Wang et al. (2020b). As illustrated in Table 5, the SQL query can be represented as a tree using the rule sequence. All of which are production rules of the SQL grammar. By exploiting every rule of the grammar, we can generate SQL queries covering patterns of different complexity.

A.3 Example Data

The response of more than 2 rows of table in CoSQL will degenerate into a template response as shown in Figure 8a. Differently, we ask the annotator to write all the input information in the response. As shown in Figure 8b, the execution result table is fully described in `ResponseNLG`. Which is more in line with real TableQA scenarios.

A.4 PARENT Metric

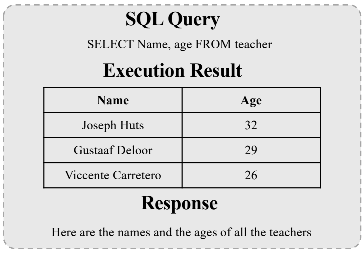
PARENT (Dhingra et al., 2019) is a metric proposed specifically for Data-to-Text generation to evaluate the faithfulness of the generated texts. It takes the input graph or table into account. However, it cannot be directly applied to TableQA Response Generation because it does not consider the input SQL. To solve this problem, we convert each condition in the SQL into a `<attribute, value>` pair. Similarly, we also convert each cell in the input table into a `<atrribute, value>` where `attribute` is the column name and `value` is the cell value. Figure 9 shows an example. However, according to our human evaluation, this metric is imperfect and does not consider the relation between attribute and value in each pair, especially the pair from the SQL. The modified code will be released with our dataset.

A.5 Case Study

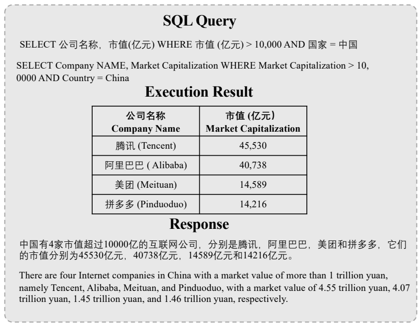
Figure 10 shows an example case generated by FINETUNE and our final model. The result table has four rows and two columns, and the FINE-

SQL Production Rules
`SQLs ::= SQL	SQL interaction SQLs	SQL union SQLs	...`
`SQL ::= Select	Select Where	Select Order	Select Order Filter`
`Select ::= Select A	Select AA	...`
`Where ::= Where Conditions`
`Conditions ::= A op value	A op SQL`
`A ::= C	MIN C	MAX C	AVG C	COUNT C	SUM C`
`C ::= table.column`
`op ::= ==	!=	>	>=	<	<=`

Table 5: SQL generation grammar rules.



(a) Example from CoSQL



(b) Example from ResponseNLG

Figure 8: Example data from CoSQL and ResponseNLG.

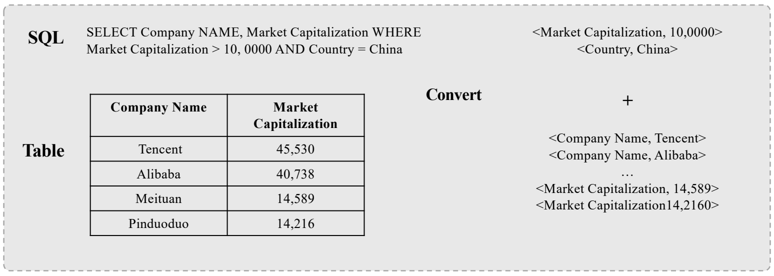


Figure 9: Example case of PARTENT metric.

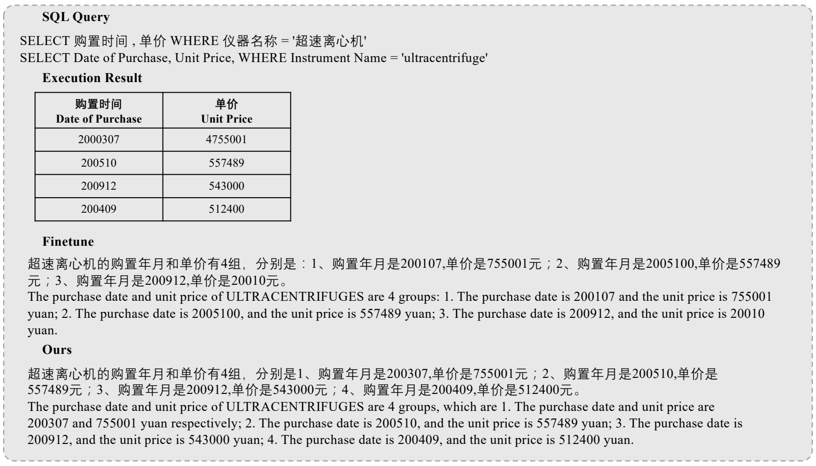


Figure 10: Example case of different generation models

TUNE model only describe three rows of the results. Differently, our model can be completely faithful to a given SQL query and table, even for relatively large data.