

CATS: A Pragmatic Chinese Answer-to-Sequence Dataset with Large Scale and High Quality

Liang Li^{1,2}, Ruiying Geng³, Chengyang Fang^{1,2}, Bing Li¹, Can Ma^{1*},
Rongyu Cao³, Binhua Li³, Fei Huang³, Yongbin Li^{3*}

¹Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

²School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

³DAMO Academy, Alibaba Group

{liliang, macan}@iie.ac.cn

{ruiying.gry, shuide.lyb}@alibaba-inc.com

Abstract

There are three problems existing in the popular data-to-text datasets. First, the large-scale datasets either contain noise or lack real application scenarios. Second, the datasets close to real applications are relatively small in size. Last, current datasets bias in the English language while leaving other languages underexplored. To alleviate these limitations, in this paper, we present CATS, a pragmatic Chinese answer-to-sequence dataset with large scale and high quality. The dataset aims to generate textual descriptions for the answer in the practical TableQA system. Further, to bridge the structural gap between the input SQL and table and establish better semantic alignments, we propose a Unified Graph Transformation approach to establish a joint encoding space for the two hybrid knowledge resources and convert this task to a graph-to-text problem. The experiment results demonstrate the effectiveness of our proposed method. Further analysis on CATS¹ attests to both the high quality and challenges of the dataset.

1 Introduction

Data-to-text (D2T) generation (Kukich, 1983; Reiter and Dale, 1997) aims to generate a natural language description conditioned on structured or semi-structured data, such as graphs (Song et al., 2018; Wang et al., 2020c) or tables (Lebret et al., 2016; Wiseman et al., 2017). It helps people get the key points of the input data and makes the stored information accessible to a broader range of end-users. A large number of datasets have been proposed as the testbed for neural D2T models and are driving the domain.

However, as shown in Table 1, we note three problems existing in the popular datasets. First, the large-scale datasets either contain noises (e.g.,

*Corresponding authors: Can Ma, Yongbin Li

¹CATS is available at <https://github.com/AlibabaResearch/DAMO-ConvAI/tree/main/cats>

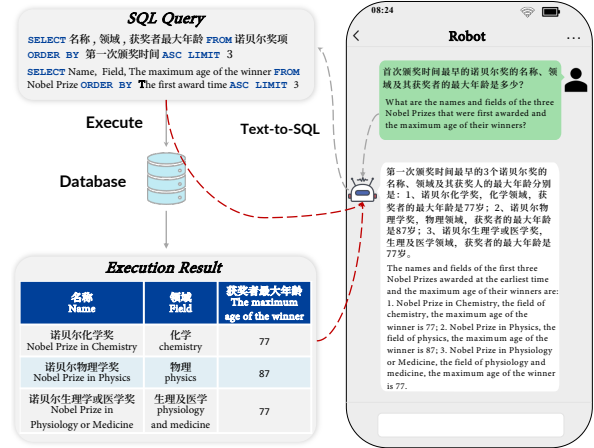


Figure 1: An example for a practical TableQA system. The red dotted lines denote the input data for answer-to-sequence.

WEATHERGOV (Liang et al., 2009)) or lack practical application scenarios, e.g., ToTTo (Parikh et al., 2020). The shortcoming leads to a separation between research and application. Second, the datasets close to practical scenarios are relatively small in size. For example, ROTOWIRE (Wiseman et al., 2017) only contains 4.9K training examples, and CoSQL (Yu et al., 2019) is consist of 7.8K training pairs. The small training size can easily lead to overfitting and is not conducive to training a reliable neural network model. Lastly, most of the existing datasets are built for English, which leads to advanced work on D2T generation primarily focusing on English and leaving other languages underexplored. These limitations hinder the progress of D2T generation. We therefore need to investigate possible remedies.

The crucial step to improving the above limitations is digging out a data-to-text task with a practical scenario. Recently, CoSQL (Yu et al., 2019) has proposed a practical controlled D2T task: answer-to-sequence. As shown in Figure 1, the task takes a SQL query generated by a semantic parsing module, i.e., text-to-SQL (Zettlemoyer and Collins,

2012), and its corresponding execution result (in the form of a table) as the model input and aims to produce a natural language description as the response to users in a real-world TableQA system. The SQL gives explicit signals for models on what to generate. The generated description could provide a concise and easy-to-understand summary of the result table and help users verify whether the queried result is consistent with the original question (Fang et al., 2022). Moreover, the task also contributes to a more user-friendly human-computer interaction. Nevertheless, CoSQL contains only 7.8K answer-to-sequence examples for training. Additionally, it is a dataset with SQL-grounded dialogue state tracking as the core, and the generation annotations are very rough. The scale and quality of CoSQL limit further exploring the answer-to-sequence task.

In this paper, to bridge the gap between research and application of data-to-text datasets and enrich their language diversity, we comply with the CoSQL setting and present CATS, a large-scale and high-quality Chinese answer-to-sequence dataset. We manually annotate all collected SQL-table pairs to obtain their descriptions. We make two efforts to improve the quality and scale of the collected SQL-Table pairs and guarantee they are close to practical scenarios. First, we annotate the SQL-table pairs from DuSQL (Wang et al., 2020b), a large-scale Chinese Text-to-SQL dataset with a SQL query distribution close to real applications. Data collected in this way are named CATS-D. Second, we adopt an automatic data construction pipeline to collect a large number of SQL-table pairs for annotation. The basic idea is automatically crawling a mount of tables from the Internet to build multi-table databases and then automatically generating SQL queries based on the SQL grammar and constrained by the given database. Data collected with this method are referred to as CATS-S. Compared to CATS-D, CATS-S expands the data scale while reducing the share of easy SQLs to make the dataset more challenging. In total, CATS is made up of both CATS-D and CATS-S, and contains 43,369 answer-to-sequence examples, which is an order of magnitude larger than CoSQL.

The input SQL and table in answer-to-sequence are heterogeneous, and there is a structural gap between them. To bridge the gap and establish better semantic alignments, we propose a Unified Graph Transformation approach (UGT), which first

converts the two sources to two undirected graphs, then builds the connection between the nodes in different graphs to obtain a unified graph. In this way, we convert this task to a graph-to-text problem (Gardent et al., 2017b). Previous graph-to-text work (Ribeiro et al., 2021) transforms the input graph into a new token graph to apply pretrained language models, such as T5 (Raffel et al., 2020). We consider that this transformation breaks the original input graph structure and may bring in extra noises into graph encoding. Hence, we further introduce a Node Segment Embedding (NSE) to preserve original structure information.

Our contributions are three-fold as follows:

- We present a large-scale and high-quality Chinese answer-to-sequence dataset (CATS), which narrows the gap between research and application of data-to-text generation datasets and enriches the language diversity.
- We propose UGT and NSE to better model the input of two heterogeneous structured input data sources.
- Experiments and analysis on CATS attest to both the high quality and challenges of the dataset. The results also demonstrate the effectiveness of our proposed method.

2 Related Works

2.1 Answer-to-Sequence Generation

In a real-world setting, a TableQA system comprises a table semantic parsing (text-to-SQL) component and an answer-to-sequence component. The semantic parsing component converts a natural language question into a SQL query (Guo et al., 2019; Wang et al., 2020a; Hui et al., 2021) and the answer-to-sequence component aims generating a natural language description of the SQL and the execution result. CoSQL (Yu et al., 2019) first proposes the answer-to-response task and refers to it as response generation. Intuitively, response generation should encompass both answer acquisition and answer description, which could easily be confused with the role of the whole Table QA system. Therefore, to make the task more clearly related to its definition and function, we rename it as answer-to-sequence generation. In this paper, the proposed CATS follows the same task setting in CoSQL. Specifically, the task’s input consists of a SQL query and its corresponding execution result (in the form of a table), and the output is a natural language description.

Dataset	Train Size	Domain	Target	Application	Language
WEATHERGOV (Liang et al., 2009)	25K	Weather	Crawled	Weather Report	English
WikiBio (Lebret et al., 2016)	583K	Wikipedia	Crawled	-	English
WebNLG (Gardent et al., 2017a)	25.3K	DBPedia	Annotated	-	English
LogicNLG (Chen et al., 2020)	28.5K	Wikipedia	Annotated	-	English
ToTTo (Parikh et al., 2020)	120K	Wikipedia	Annotated	-	English
Rotowire (Wiseman et al., 2017)	4.9K	NBA	Annotated (Noisy)	NBA	English
AdverGeneration (Shao et al., 2019)	115K	Chinese E-commerce	Crawled	Advertising Text Generation	Chinese
CoSQL (Yu et al., 2019)	7.8K	Cross-Domain	Annotated	TableQA	English
Map2seq (Schumann and Riezler, 2021)	7.6K	OpenStreetMap	Annotated	Navigation	English
CATS	34.7K	Cross-Domain	Annotated	TableQA	Chinese
CATS-D	6.7K	Cross-Domain	Annotated	TableQA	Chinese
CATS-S	26.4K	Cross-Domain	Annotated	TableQA	Chinese

Table 1: Comparison of popular data-to-text datasets in different aspects. **Application** represents the practical application scenario associated with the dataset.

Especially, using SQL query as input rather than natural language question is more practical in multi-turn TableQA scenarios because the SQL query can easily represent the context state (Yu et al., 2019).

2.2 Structure Modeling in Data-to-Text

Recently, some works in D2T generation have shown that the structure modeling for the input data can dramatically improve the model performance. For table data, Liu et al. (2019); Li et al. (2021) propose to utilize a hierarchical encoder to model the table’s representation from the row and column levels. For graph structure modeling, early works (Song et al., 2018; Damonte and Cohen, 2019) introduce Graph Neural Networks as the structure encoder, which only considers the relations between neighbor nodes. Unlike the local encoding strategies, Zhu et al. (2019); Cai and Lam (2020) propose the Graph Transformer that uses explicit relation encoding and allows direct communication between two distant nodes. Newly, some works enable the pretrained language models the structure modeling capabilities and achieve SOTA results on many D2T tasks. Especially, Ribeiro et al. (2021) attempt to insert structural adapters into T5’s encoder to model the graph structure. Wang et al. (2022) modify the T5’s attention masking matrix to encode table with a structure-aware self-attention mechanism. In this paper, we propose to utilize UGT to convert the input SQL and table to a graph and utilize a graph-to-model to model it. Our model refers to Ribeiro et al. (2020b, 2021)’ works and further improves them by introducing NSE to better preserve the graph structure.

3 Dataset Construction

Considering the limitations of existing D2T datasets, we present CATS, a massive and pragmatic Chinese answer-to-sequence dataset. CATS is constructed by two phases: SQL-table pairs collection and manual data annotation. To balance the data quality and scale and bring it closer to the practical scenario, we collect the SQL-table pairs in two ways. First, we derive SQL-table pairs from DuSQL (Wang et al., 2020b), a text-to-SQL dataset that generates the SQL queries by referring to the SQL query distribution in real-life applications. The dataset obtained by annotating these pairs is referred to as CATS-D. Besides, we implement an automatic data construction pipeline to collect massive high-quality SQL-table pairs. Data collected with this method are referred to as CATS-S, which increases the proportion of complicated SQL queries to make the dataset more challenging. Ultimately, both CATS-D and CATS-S make up CATS. We first describe how to obtain SQL-table pairs for subsequent annotation and then introduce the annotation details.

Database Building To mimic the practical TableQA system, we first follow Wang et al. (2020b) to build a multi-table database D^d by collecting all databases in DuSQL. In addition, we also build another multi-table database D^s for expanding the size and domain of our dataset through a table collection pipeline. Specifically, 100,000 high-frequency words are first summarized from the CLUE (Xu et al., 2020) corpus. Then, we query these words in Google and download all the queried spreadsheet files. Subsequently, the available tables in these spreadsheets are extracted by a table parser that can identify the potential table in a work-

sheet. To protect personal privacy, we use predefined unique words to replace sensitive information in these tables, such as passwords, ID numbers, credit card numbers, etc. Finally, these tables are used to construct the database D^s . Please refer to Appendix A.1 for more details.

SQL and Table Collection We execute all the SQL queries in DuSQL in the database D^d to get their corresponding tables. This is consistent with how a practical Table QA system answers user questions after parsing it to SQL. Then we discard SQL-table pairs containing SQLs that execute with empty results to obtain a SQL-table pair set $\text{CATS-}D^{un} = \{s_i^d, t_i^d\}_{i=1}^n$. DuSQL does not release the code for generating synthetic queries. Therefore, to increase the annotation examples, we reimplement a SQL generator similar to the one in DuSQL. Notably, the generated SQL contains both single-table and multi-table queries. Please refer to Appendix A.2 for more detailed information on the SQL generator. The sampled SQLs which cannot execute in database D^s or execute with empty results are deserted. In this way, we obtain another SQL-table pair set $\text{CATS-}S^{un} = \{s_i^s, t_i^s\}_{i=1}^m$.

Data Annotation Process We employ 20 well-educated crowd workers to annotate the SQL-table pairs in $\text{CATS-}D^{un}$ and $\text{CATS-}S^{un}$. In particular, the annotators are asked to write a description y given a SQL s and table t pair. They must follow the requirements: (1) avoiding template-like language and trying to write a natural, fluent, and grammatically correct description; (2) the description must summarize all the content in the table; (3) the description must be logically consistent with the input SQL; (4) filtering the incomprehensible examples that are semantically unclear. Furthermore, 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. Finally, the annotated $\text{CATS-}D^{un}$ is named as CATS-D . To guarantee data consistency, we sample a subset from the annotated $\text{CATS-}S^{un}$ following a similar complexity distribution with CATS-D . We name the sampled dataset CATS-S . However, we find that easy SQL queries account for a large-scale proportion (**47.87%**) in CATS-D . Therefore, we reduce the proportion of easy SQLs (**14.50%**) in CATS-S to make it more challenging.

COLUMN NUMBER	1	2	3	>=4
CoSQL	6,329	1057	459	0
CATS	8,966	20,862	3242	1627
CATS-D	2,883	2,977	820	0
CATS-S	6,157	17,813	2,394	1,653
ROW NUMBER	1	2	3	>=4
CoSQL	4740	610	2,495	0
CATS	14,909	6,158	3,671	9,959
CATS-D	2,123	656	1,129	2,772
CATS-S	12,754	5,538	2,510	7,215
SQL HARDNESS	Easy	Medium	Hard	Extra Hard
CoSQL	2,788	1,826	1,717	1,514
CATS	7,223	13,000	12,016	2,458
CATS-D	3,198	1709	1,264	509
CATS-S	4,063	11,214	10,787	1,953
TARGET LENGTH	< 20	< 40	< 60	>= 60
CoSQL	7,005	825	15	0
CATS	10,319	12,862	5,864	5,652
CATS-D	1,893	2,026	1,912	849
CATS-S	8,401	10,873	3,962	4,781

Table 2: Complexity distribution comparison between CATS and CoSQL.

3.1 Dataset Analysis

The final CATS contains 43,369 examples, including 8,350 examples in CATS-D and 33,019 examples in CATS-S . Each annotated example contains a triple of SQL s , table t , and descriptive sentences y . We split the training/development/test sets by 34,697/4,336/4,336 randomly. To understand the characteristics of the data collected in CATS-D and CATS-S , we also split them accordingly. The training, development, and test sets of CATS-D and CATS-S contain 6,680/835/835 and 28,017/3,501/3,501 examples, respectively.

Data Complexity To better understand our dataset, we compare its complexity with CoSQL in four dimensions, including the input tables’ row and column number, SQL hardness, and the target length. Following Guo et al. (2021), we adopt SQL hardness to measure the complexity of SQL queries from the following four-level: easy, medium, hard, and extra hard, according to the number of components, selections, and conditions in a SQL query (Yu et al., 2018). Considering CoSQL only release the training and development sets, we only show the training set comparison. The results are summarized in Table 2. First, we find that the tables in CoSQL are small, such as 60% of the tables with only one row and more than 80% with only one column. Second, we notice that most of the descriptions in CoSQL are less than 20 in length. The first reason is that most of the input tables are small.

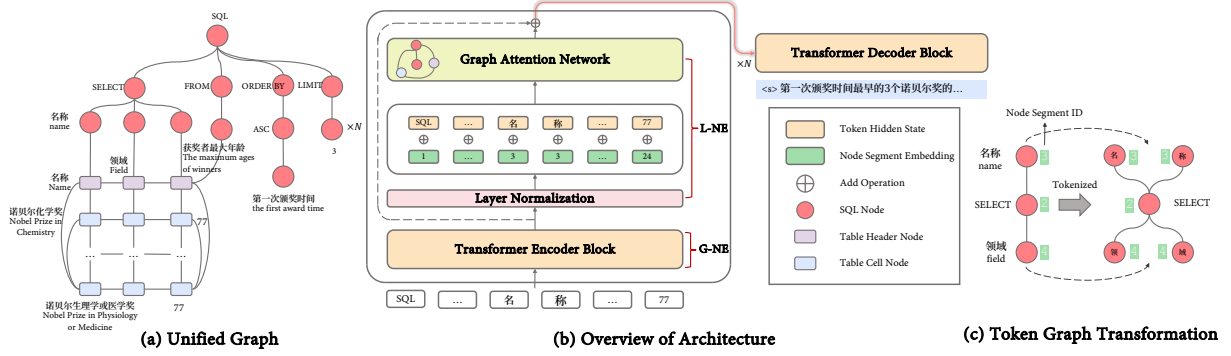


Figure 2: Illustration of the proposed method. (a) is an example of a unified graph transformed by Unified Graph Transformation. (b) is an overview of our model. **L-NE** and **G-NE** denote Local Node Encoder and Global Node Encoder, respectively. (c) is an example of token graph transformation.

By manually checking the data in CoSQL, we find the second reason is that CoSQ describes the table with more than two rows through a generic template, such as “Here are the ...”. Last, we observe that easy SQL queries in CoSQL account for **35.54%**, far more than **20.84%** in CATS. These features make CoSQL only suitable for simple scenarios and less challenging. By contrast, CATS has a broader distribution than CoSQL, which is more in line with real TableQA applications.

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 **Unified Graph Transformation** approach (UGT), which explicitly connects the input SQL and table in a unified structure. In this way, we can obtain a joint graph representation of the two sources and convert the answer-to-sequence task to a graph-to-text problem. And then, we utilize a variational transformer architecture (Ribeiro et al., 2020b) 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 2 (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 Unified Graph Transformation and the Local Node Encoder in detail.

4.1 Unified Graph Transformation

Given a SQL s and its execution result (in the form of a table) t as input (shown in Figure 1), the Unified Graph Transformation takes two steps to transform the input two sources of data into a unified graph (shown in Figure 2 (a)). 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. 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 graph. we also add a self-loop connection for each node. The transformed unified 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 2 (a) shows an example of the transformed unified graph.

We expect that developing generation model should benefit from the recent advance on pre-trained language models (PLMs). Following previous work (Ribeiro et al., 2021), 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 \mathcal{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 2 (c)). 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 noise 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 unified graph \mathcal{G}_h . Figure 2 (c) gives an example.

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 (L-NE) works. As shown in Figure 2 (b), L-NE consists of two main modules: a Node Embedding Layer and a Graph Attention Network (GAT) (Velickovic et al., 2018) Layer. The former enriches the features of the nodes, and the latter explicitly models the graph structure. Formally, given h_v , we obtain the feature-enhanced node representation by:

$$h_v^e = \text{LayerNorm}(h_v) + e_v^s, \quad (1)$$

where *LayerNorm* represents layer normalization (Ba et al., 2016). e_v^s denote the node segment embedding for node v . After the Node Embedding Layer, we utilize a GAT layer to model the graph structure. Formally, it aggregates the representations of node v in a multi-head self-attention layer (Vaswani et al., 2017) as follows:

$$\begin{aligned} s_{v,n} &= \frac{h_v^e W_Q^h (h_n^e W_K^h)^\top}{\sqrt{d/H}}, \\ \alpha_{v,n}^h &= \frac{e^{s_{v,n}^h}}{\sum_{\tilde{n} \in \mathcal{N}(v)} e^{s_{v,\tilde{n}}^h}}, \\ z^h &= \sum_{n \in \mathcal{N}(v)} \alpha_{v,n}^h (h_n^e W_V^h), \\ h^r &= \text{Concat}(z^1, \dots, z^H), \end{aligned} \quad (2)$$

where $1 \leq h \leq H$, and $W_Q^h, W_K^h, W_V^h \in \mathbb{R}^{d \times (d/H)}$. $\mathcal{N}(v)$ denotes the immediate neighborhood of node v in graph \mathcal{G} .

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:

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

5 Experiment

5.1 Experiment Settings

Baselines Due to current datasets bias in the English language, the D2T methods for others are

SQL Components	Descriptions
Min	最小的 (minimum)
Max	最大的 (maximum)
Count	数量 (the number of)
Sum	总共 (total)
Average	平均 (average)
=	等于 (is)
!=	不等于 (is not)
>	大于 (more than)
>=	大于等于 (no less than)
<	小于 (less than)
<=	不小于 (no more than)
And	并且 (and)
Or	或者 (or)
Asc	从低到高 (in the ascending)
Desc	从高到低 (in the descending)

Table 3: Natural language descriptions for different SQL components.

rarely explored. Meanwhile, PLMs-based models, such as T5, have achieved SOTA results (Ribeiro et al., 2020a, 2021; Wang et al., 2022; Jolly et al., 2022) on many D2T tasks. Therefore, we experiment with T5-based models to understand their performance on CATS-D, CATS-S, and CATS:

- TEMP automatically generates descriptions based on the predefined template. Specifically, we first manually write a template for SQL queries replacing the values, columns, table names, and conditions with slots. Meanwhile, we also create a list of descriptions for each component in SQL queries (Table 3 reports the descriptions of partial SQL components). Then we enumerate all cells in the table row by row to obtain the description for a table. Lastly, we join the two parts of descriptions as the final output.
- POINTER-GEN is an RNN-based Seq2Seq model with attention and copy mechanism (See et al., 2017). We concatenate the SQL and linearized table as input.
- T5 denotes finetuning the T5 model on the proposed CATS. The input is the same as that used in the POINTER-GEN. Notably, to make a fair comparison with our proposed method, we add a fully connected feed-forward network (FNN) on top of each transformer layer and make its parameters equal with the L-NE layer. We denote this as T5 + FNN.
- T5-GRAPH is also a finetuning T5 method. Different from T5, it uses the sample graph

MODELS	CATS			CATS-D			CATS-S		
	BLEU	ROUGE-L	COVERAGE	BLEU	ROUGE-L	COVERAGE	BLEU	ROUGE-L	COVERAGE
<i>Development</i>									
GOLD	-	-	75.56	-	-	69.59	-	-	77.30
TEMP	40.04	57.20	81.48	18.05	47.37	77.93	42.71	59.82	83.24
POINTER-GEN	51.26 \pm 0.20	73.70 \pm 0.14	68.73 \pm 0.13	48.33 \pm 0.91	67.95 \pm 0.96	56.96 \pm 0.90	49.77 \pm 0.16	73.79 \pm 0.26	69.26 \pm 0.24
T5	53.60 \pm 0.13	74.42 \pm 0.06	72.87 \pm 0.04	52.47 \pm 0.28	68.5 \pm 0.32	68.20 \pm 0.25	51.43 \pm 0.10	73.77 \pm 0.04	73.08 \pm 0.03
T5 + FNN	54.14 \pm 0.21	74.80 \pm 0.16	72.85 \pm 0.18	52.10 \pm 0.17	68.28 \pm 0.17	68.02 \pm 0.31	51.67 \pm 0.22	73.75 \pm 0.17	73.08 \pm 0.17
T5-GRAPH	52.21 \pm 0.17	73.68 \pm 0.04	72.03 \pm 0.10	49.89 \pm 0.40	66.72 \pm 0.10	66.65 \pm 0.26	50.12 \pm 0.18	73.11 \pm 0.13	72.05 \pm 0.04
T5-GRAPH + FNN	52.30 \pm 0.17	73.71 \pm 0.20	71.87 \pm 0.05	48.81 \pm 0.27	66.35 \pm 0.13	66.10 \pm 0.30	50.42 \pm 0.09	73.22 \pm 0.12	72.07 \pm 0.05
UGT	54.75 \pm 0.15	75.72 \pm 0.06	72.68 \pm 0.16	54.23 \pm 0.49	69.82 \pm 0.35	68.07 \pm 0.63	52.54 \pm 0.16	74.84 \pm 0.12	72.99 \pm 0.07
UGT + NSE	56.34 \pm 0.13	76.72 \pm 0.09	73.41 \pm 0.05	58.79 \pm 0.51	73.16 \pm 0.31	68.94 \pm 0.31	53.54 \pm 0.15	75.36 \pm 0.19	73.67 \pm 0.10
<i>Test</i>									
GOLD	-	-	76.35	-	-	68.67	-	-	76.98
TEMP	41.39	57.82	82.40	17.76	46.21	77.83	42.69	60.16	82.96
POINTER-GEN	50.77 \pm 0.56	73.25 \pm 0.14	68.47 \pm 0.31	47.34 \pm 0.81	66.46 \pm 0.80	56.93 \pm 1.21	50.37 \pm 0.27	74.21 \pm 0.20	69.98 \pm 0.24
T5	53.49 \pm 0.13	74.22 \pm 0.08	72.36 \pm 0.12	51.32 \pm 0.22	66.81 \pm 0.28	67.93 \pm 0.18	52.91 \pm 0.07	74.51 \pm 0.08	73.33 \pm 0.08
T5 + FNN	53.87 \pm 0.18	74.42 \pm 0.16	72.34 \pm 0.10	50.71 \pm 0.12	66.42 \pm 0.24	67.06 \pm 0.24	52.71 \pm 0.14	74.32 \pm 0.11	73.32 \pm 0.16
T5-GRAPH	51.82 \pm 0.13	73.28 \pm 0.05	71.33 \pm 0.03	47.91 \pm 0.28	64.75 \pm 0.20	65.51 \pm 0.31	51.40 \pm 0.22	73.78 \pm 0.13	72.15 \pm 0.08
T5-GRAPH + FNN	52.04 \pm 0.22	73.58 \pm 0.15	71.37 \pm 0.13	47.45 \pm 0.33	64.60 \pm 0.25	65.69 \pm 0.31	51.35 \pm 0.21	78.78 \pm 0.14	72.32 \pm 0.12
UGT	54.27 \pm 0.24	75.13 \pm 0.10	72.13 \pm 0.16	52.48 \pm 0.43	67.96 \pm 0.45	67.19 \pm 0.72	53.03 \pm 0.37	75.38 \pm 0.11	73.18 \pm 0.13
UGT + NSE	55.95 \pm 0.23	76.10 \pm 0.06	72.84 \pm 0.18	57.10 \pm 0.42	71.74 \pm 0.43	68.40 \pm 0.23	54.21 \pm 0.17	75.93 \pm 0.20	74.04 \pm 0.08

Table 4: Automatic evaluation results on the development and test sets. Mean (\pm s.d.) over 4 seeds.

representation with our method (described in Section 4.1) as input. Again, we add FNN to make a fair comparison, which is denoted as T5-GRAPH + FNN.

Evaluation Metrics We evaluate our models by applying both automatic and human evaluations. For automatic evaluation, we employ the widely used metric, BLEU (Papineni et al., 2002) and ROUGE-L (Lin, 2004), to evaluate the fluency of generated text. And we utilize SacreBLEU (Post, 2018) to calculate the BLEU after segmenting the sentence by jieba². Additionally, we utilize COVERAGE (Shao et al., 2019) to evaluate the faithfulness of generated text. COVERAGE measures the average proportion of input tables that are covered by a generated text. The table headers are also considered. We use string matching rules to determine whether a cell exists in the generated text. We conduct experiments over 4 different seeds and report the average scores on them.

We display examples of input representation for different models and provide the implementation details in Appendix C.1 and C.2.

5.2 Main Result

Table 4 presents the experimental results on CATS, CATS-D, and CATS-S, from which we make three main observations.

²<http://pypi.python.org/pypi/jieba>

First, we can see that all neural network models outperform TEMP on BLEU by a large margin. This suggests that neural models are better at generating fluent expressions. We consider this thanks to the language modeling task (Equation 3), which trains the neural models to predict the next token, given the previous history. Nevertheless, we find that TEMP achieves the best COVERAGE scores on all sets, even better than GOLD. We consider this is because, when annotating the references, to make the presentation more reasonable and fluent, annotators summarize the contents of the table, such as merging some cells, etc. On the other hand, TEMP copies all the contents of the table directly.

Second, adding extra trainable parameters (+ FNN) does not always improve the performance on T5 and T5-GRAPH. For example, T5 + FNN performs better than T5 on both CATS and CATS-S, but worse than T5 on CATS-D. Moreover, we notice that T5 performs better than T5-GRAPH given the fact that the sizes of their parameters are equal. We speculate this is because, compared to T5-GRAPH, T5 uses the original SQL and the flattened table as input, which preserves the partial structural information of the input SQL and table by the segment symbols “,” and “|” (please refer to Appendix C.1 for the example of input data linearizations). However, T5-GRAPH still treats the input as a sequence and ignores the unified graph’s structure, leading to its performance degradation.

MODEL	CATS	CATS-D	CATS-S
T5 + FNN	54.14 \pm 0.21	52.10 \pm 0.17	51.67 \pm 0.22
w/o SQL	40.90 \pm 0.24	39.75 \pm 0.08	40.00 \pm 0.30
w/o TABLE	17.83 \pm 0.13	24.25 \pm 0.33	14.51 \pm 0.11
OURS	56.34 \pm 0.13	58.79 \pm 0.51	53.54 \pm 0.15
w/o SQL	45.16 \pm 0.26	47.92 \pm 0.50	43.89 \pm 0.38
w/o TABLE	19.59 \pm 0.16	26.91 \pm 0.11	16.20 \pm 0.62

Table 5: Effect of input SQL and TABLE. w/o SQL and w/o TABLE denote removing the SQL and table from the input, respectively. OURS denotes UGT + NSE.

Lastly, by explicitly modeling the unified graph structures, UGT dramatically outperforms the size-comparable models T5-GRAPH + FNN and T5-FNN on all metrics. The results display UGT’s superiority in capturing essential structural knowledge for this task. Additionally, Node Segment Embedding (+ NSE) further improves the performance. This verifies that NSE can help the encoder better preserve the original structural information.

5.3 Analysis and Discussion

Effects of input SQL and Table To examine the effects of different input data, we conduct ablation studies on the input side by removing the input SQL and table. The results on three development sets are summarized in Table 5. We observe that, after removing the SQL and only utilizing the table as input, both T5 + FNN and our method (UGT + NSE) perform poorly on all metrics. The performance degrades even more if only SQL is employed. The results demonstrate that both input SQL and table are essential for the answer-to-sequence task. Additionally, our method clearly outperforms T5 + FNN on all ablation settings. It reveals the effectiveness of our method compared to vanilla T5 architecture even under extreme input conditions.

Effects of Data Complexity We further explore the performances on different levels of data complexity. We use BLEU as the metric in this section. The results are shown in Table 6. We first explore the effect of the table size. Unsurprisingly, the BLEU scores of all models decrease as the number of table rows or columns grows. The more rows or columns the table contains, the more difficult it is for a model to process it. Compared to two baseline models, our method is better at handling large tables. Furthermore, we investigate the impact of SQL complexity on model performances. With respect to the SQL complexity, our model

COLUMN NUMBER	1	2	3	≥ 4
# EXAMPLES	1,138	2,580	403	215
POINTER-GEN	53.21	50.74	42.20	35.29
T5 + FNN	+2.28	+1.16	+7.08	+4.29
OURS	+5.61	+4.69	+7.54	+5.28
ROW NUMBER	1	2	3	≥ 4
# EXAMPLES	1,899	769	467	1201
POINTER-GEN	56.72	49.71	49.05	44.30
T5 + FNN	+3.57	-0.58	+1.68	+6.24
OURS	+5.75	+1.54	+5.16	+7.62
SQL HARDNESS	Easy	Medium	Hard	Extra Hard
# EXAMPLES	915	1,588	1,531	302
POINTER-GEN	60.92	54.99	42.78	43.17
T5 + FNN	+0.92	+0.60	+6.79	+3.65
OURS	+3.98	+3.75	+7.80	+9.22
TARGET LENGTH	< 20	< 40	< 60	≥ 60
# EXAMPLES	1,275	1,635	724	702
POINTER-GEN	52.67	51.97	52.02	41.64
T5 + FNN	+2.93	-0.31	-0.06	+7.54
OURS	+6.08	+3.19	+3.33	+7.82

Table 6: BLEU scores of different models in the CATS test set on different levels of data complexity. Relative results of T5 + FNN and our method are reported compared against the POINTER-GEN.

achieves larger improvement against baseline models, especially on extra hard SQLs. It shows that our approach can better encode the complex input data than others. Lastly, we study the model performance concerning different ground-truth description lengths. The POINTER-GEN struggles on longer descriptions, where the performance drops over 10 BLEU scores on responses longer than 60. In this scenario, T5-based models dramatically outperform the POINTER-GEN, while our method can still beat T5 + FNN.

5.4 Human Evaluation

To reach a deeper understanding of the qualities of the generated descriptions, we conduct human evaluation following Parikh et al. (2020). We compare our method with TEMP, POINTER-GEN, and T5 + FNN. Specifically, we first randomly select 100 examples from the CATS test set and the corresponding outputs generated by each model. And then, five native Chinese annotators (three females and two males) with master’s degrees or above engaged in NLP research are invited to evaluate the quality from the four axes. Specifically, **FLUENCY** measures whether the description is fluent. **FAITHFULNESS** estimates whether the description is logically consistent with input SQL, and all pieces of information are supported by the input table.

MODEL	Flu. \uparrow	Fai. \uparrow	Cov.(%) \uparrow	Rep. \downarrow
GOLD	8.42	9.15	95.32	0.14
TEMP	5.27	6.87	99.41	0.02
POINTER-GEN	6.13	6.32	83.27	0.74
T5 + FNN	6.82	7.16	89.27	0.39
OURS	7.14	7.48	90.26	0.27

Table 7: Human evaluation over references (denoted as GOLD) and model outputs. Flu., Fai., Cov., Rep. denote FLUENCY, FAITHFULNESS, COVERAGE and REPETITION. \uparrow indicates higher is better and \downarrow denotes lower is better.

They are scores range from 1 to 10, the higher the better. **COVERAGE** is the percentage of cells in the input table the candidate sentence covers. It is different from the one in Table 4 (please refer to Appendix C.4). **REPETITION** is number of cells the candidate sentence repeats. We also introduce the reference as one candidate (denoted as GOLD). And its results can be regarded as the upper bound.

The results summarized in Table 7 show that the GOLD consistently achieves high performance than generation methods. It attests to the high quality of our human annotations. We report FLUENCY and FAITHFULNESS score for TEMP because they are sensitive evaluation. We can see that TEMP gets a high FAITHFULNESS score but is poor on FLUENCY. Our method outperforms baseline models on almost all axes with an agreement kappa score (van der Lee et al., 2020) more than 0.86. It demonstrates the effectiveness of our proposed method. Although our model achieves a high coverage rate (90.26%), its FAITHFULNESS score is relatively low (only 7.48), and there is a considerable gap compared with GOLD. 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 deep semantics of SQL and table, which is the biggest challenge in this dataset.

6 Conclusion

We present CATS, a large-scale and high-quality Chinese answer-to-sequence dataset, along with a series of baselines. It helps alleviate the problem of current D2T datasets’ bias towards the English language. We propose a Unified Graph Transformation method to bridge the structural gap between the SQL and table. In this way, we convert the task to a graph-to-text problem. Furthermore, 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 CATS show that our proposed model outperforms existing baseline models. We conduct further analysis on CATS, which attests to both the high quality and challenges of the dataset.

Limitations

This work presents CATS, a large-scale and high-quality Chinese answer-to-sequence dataset. It is a free and open dataset. One of most important motivations for presenting this dataset is that most of the existing datasets are built for English, which leads to advanced work on D2T generation primarily focusing on English and leaving other languages underexplored. However, CATS only alleviates the dataset language bias rather than solving it. And it is limited to the study of Chinese methods. Regarding methodology, the proposed UGT converts the answer-to-sequence task to a graph-to-text problem to bridge the gap between two heterogeneous input data (SQL and table). However, UGT works only for answer-to-sequence task rather than graph-to-text task. Additionally, though the proposed NSE can help the graph-to-text model better preserve the original structural information, the contribution may be limited to the graph-to-text task.

Ethics Statement

This work presents CATS, a free and open dataset for the research community to study the answer-to-sequence problem in the practical TableQA system. And it helps enrich the D2T languages and alleviate the datasets’ bias in English. To balance the data quality and scale and bring it closer to the practical scenario, data in CATS are collected from two sources, which are manually annotated as CATS-D and CATS-S. In other words, CATS consists of CATS-D and CATS-S. The data in CATS-D is collected from DuSQL (Wang et al., 2020b) dataset, a free and open dataset for the Chinese Text-to-SQL problem. Meanwhile, to enlarge our dataset, we adopt an automatic data construction pipeline to collect a large number of high-quality SQL-table pairs for annotation. To ensure the quality of our dataset, we manually annotate the SQL-table pairs. We hire 24 native annotators with undergraduate degrees to annotate the data. Specifically, 20 annotators are responsible for annotations, and another 4 workers are asked to review the annotated data.

We pay 2.1 yuan (\$0.31 USD) for annotating each SQL-table pair.

To avoid our dataset leakages personal privacy, we replace the sensitive information in the collected tables with predefined unique words. Furthermore, we ask the annotators to filter out the examples that leak personal privacy and contain social bias and harmful content.

References

- Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. [Layer normalization](#). *CoRR*, abs/1607.06450.
- 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 *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5020–5027. AAAI Press.
- Deng Cai and Wai Lam. 2020. [Graph transformer for graph-to-sequence learning](#). In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7464–7471. AAAI Press.
- Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020. [Logical natural language generation from open-domain tables](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7929–7942. Association for Computational Linguistics.
- 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 3649–3658. Association for Computational Linguistics.
- Shineng Fang, Jiangjie Chen, Xinyao Shen, Yunwen Chen, and Yanghua Xiao. 2022. : A faithful contrastive framework for response generation in tableqa systems. In *International Conference on Database Systems for Advanced Applications*. Springer.
- Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. 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.
- Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017b. [The webnlg challenge: Generating text from RDF data](#). In *Proceedings of the 10th International Conference on Natural Language Generation, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, pages 124–133. Association for Computational Linguistics.
- Jiaqi Guo, Ziliang Si, Yu Wang, Qian Liu, Ming Fan, Jian-Guang Lou, Ziji 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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2316–2331. Association for Computational Linguistics.
- 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. Association for Computational Linguistics.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. [Long short-term memory](#). *Neural Comput.*, 9(8):1735–1780.
- 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 *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 13116–13124. AAAI Press.
- Shailza Jolly, Zi Xuan Zhang, Andreas Dengel, and Lili Mou. 2022. [Search and learn: Improving semantic coverage for data-to-text generation](#). In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelfth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 10858–10866. AAAI Press.
- Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. [OpenNMT: Open-source toolkit for neural machine translation](#). In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

- Karen Kukich. 1983. [Design of a knowledge-based report generator](#). In *21st Annual Meeting of the Association for Computational Linguistics, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, June 15-17, 1983*, pages 145–150. ACL.
- 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, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1203–1213. The Association for Computational Linguistics.
- 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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 5979–5989. Association for Computational Linguistics.
- Percy Liang, Michael I. Jordan, and Dan Klein. 2009. [Learning semantic correspondences with less supervision](#). In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 91–99. The Association for Computer Linguistics.
- Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.
- 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 *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6786–6793. AAAI Press.
- Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.
- 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, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.
- Ankur P. 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 2020, Online, November 16-20, 2020*, pages 1173–1186. Association for Computational Linguistics.
- Matt Post. 2018. [A call for clarity in reporting BLEU scores](#). In *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pages 186–191. Association for Computational Linguistics.
- 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](#). *J. Mach. Learn. Res.*, 21:140:1–140:67.
- Ehud Reiter and Robert Dale. 1997. [Building applied natural language generation systems](#). *Nat. Lang. Eng.*, 3(1):57–87.
- Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020a. [Investigating pretrained language models for graph-to-text generation](#). *CoRR*, abs/2007.08426.
- Leonardo F. R. Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020b. [Modeling global and local node contexts for text generation from knowledge graphs](#). *Trans. Assoc. Comput. Linguistics*, 8:589–604.
- Leonardo F. R. 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 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 4269–4282. Association for Computational Linguistics.
- Raphael Schumann and Stefan Riezler. 2021. [Generating landmark navigation instructions from maps as a graph-to-text problem](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 489–502. Association for Computational Linguistics.
- Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. [Get to the point: Summarization with pointer-generator networks](#). *CoRR*, abs/1704.04368.
- Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, and Xiaoyan Zhu. 2019. [Long and diverse text generation with planning-based hierarchical variational model](#). 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 3255–3266. Association for Computational Linguistics.

- Lin Feng 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, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1616–1626. Association for Computational Linguistics.
- Chris van der Lee, Albert Gatt, Emiel van Miltenburg, and Emiel Kraemer. 2020. Human evaluation of automatically generated text: Current trends and best practice guidelines. *Computer Speech & Language*, page 101151.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.
- Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. [Graph attention networks](#). In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.
- Bailin Wang, Richard Shin, Xiaodong Liu, Aleksandr Polozov, and Matthew Richardson. 2020a. [RAT-SQL: relation-aware schema encoding and linking for text-to-sql parsers](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7567–7578. Association for Computational Linguistics.
- Fei Wang, Zhewei Xu, Pedro A. Szekely, and Muhao Chen. 2022. [Robust \(controlled\) table-to-text generation with structure-aware equivariance learning](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 5037–5048. Association for Computational Linguistics.
- Lijie Wang, Ao Zhang, Kun Wu, Ke Sun, Zhenghua Li, Hua Wu, Min Zhang, and Haifeng Wang. 2020b. [Dusql: A large-scale and pragmatic chinese text-to-sql dataset](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6923–6935. Association for Computational Linguistics.
- Tianming Wang, Xiaojun Wan, and Shaowei Yao. 2020c. [Better amr-to-text generation with graph structure reconstruction](#). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3919–3925. ijcai.org.
- 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, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2253–2263. Association for Computational Linguistics.
- 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, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaowei Hua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. 2020. [CLUE: A chinese language understanding evaluation benchmark](#). In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 4762–4772. International Committee on 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 R. Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter S. Lasecki, and Dragomir R. Radev. 2019. [Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1962–1979. Association for Computational Linguistics.
- Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir 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, October 31 - November 4, 2018*, pages 3911–3921. Association for Computational Linguistics.

Luke S. Zettlemoyer and Michael Collins. 2012. [Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars](#). *CoRR*, abs/1207.1420.

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 2019, Hong Kong, China, November 3-7, 2019*, pages 5458–5467. Association for Computational Linguistics.

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 ::= T.column T.column mathop T.column
T ::= table in current database
mathop ::= + - *
op ::= = != > >= < <= like in not in

Table 8: SQL generation grammar rules.

A Dataset Construction Details

A.1 Database Building Details

To build the database, we first clean the collected tables. We build a rule-based table cleaning pipeline to guarantee table quality. We filter out noise tables via rules as follows: (1) We first build 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; (2) 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; (3) We filter out tables with less than 2 columns or rows; (4) We will filter out the table, if a value repeats more than 50% in it. Finally, we obtain 24K high-quality tables.

The original crawled data are in the form of independent tables, which need to be linked with other tables to form databases. We build a database creation pipeline and link different tables based on the header overlap (Wang et al., 2020b) to acquire multi-table databases. Finally, 600 databases are selected in the dataset.

A.2 Automatic SQL Generator

The SQL generator utilizes production rules from the SQL grammar to automatically generate SQL queries. Specifically, a SQL query can be represented as an abstract syntax tree (AST) 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. By exploiting every rule of the grammar, we can generate SQL queries covering patterns of different complexity along with

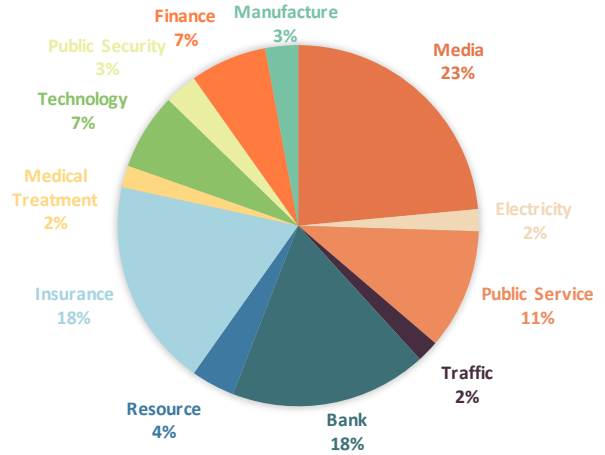


Figure 3: Topic distribution of CATS.

corresponding tables. We illustrate some SQL production rules in Table 8.

A.3 SQL Hardness

Following Guo et al. (2021), we adopt SQL hardness to measure the complexity of SQL queries from the following four-level: easy, medium, hard, and extra hard (Yu et al., 2018). The SQL difficulty is defined based on the number of SQL components, selections, and conditions. Therefore, queries that contain more SQL keywords (GROUP BY, ORDER BY, INTERSECT, nested subqueries, column selections, and aggregators, etc.) are considered harder. For example, a query is considered hard if it includes more than two SELECT columns, more than two WHERE conditions, and GROUP BY two columns, or contains EXCEPT or nested queries. A SQL with more additions on top of that is considered extra hard.

B Topics Distribution of CATS

Following Parikh et al. (2020), we build a topic categorization model for tables in CATS to investigate the topic distribution. We first ask the annotators to label 10,000 tables and then train a table topic classifier built on a table-aware encoder (Bao et al., 2018). We apply the classifier to label other table topics. Figure 3 presents an aggregated topic analysis of our dataset. We find that 61% of CATS is made up of the Media, Insurance, and Bank topics, and the other 39% is composed of broader topics, such as Public Service, Technology, and Finance. The proposed CATS is limited to topics that are presented in CLUE and DuSQL.

C Experimental Details

C.1 Example of SQL and Table Linearizations

We display the input representations for different models in Figure 4. For POINTER-GEN, T5, and T5 + FNN, we directly concatenate the SQL and linearized table as input, where table is linearized row by row. For T5-GRAPH, T5-GRAPH + FNN and OURS, follow previous work (Ribeiro et al., 2021), we linearize the SQL graph \mathcal{G}_s into a sequence of nodes by the depth-first traversal and concatenate it with the linearized table as input. Especially, instead of segmenting the nodes with special symbol $|$, we build a connection matrix for the token graph \mathcal{G} . The connection matrix is used by the Local Node Encoder to encoding the graph structure.

C.2 Implementation Details

We employ the POINTER-GEN implemented by OpenNMT (Klein et al., 2017). POINTER-GEN is built based on LSTM (Hochreiter and Schmidhuber, 1997). We set the layers of the encoder and decoder as 2 and 1, respectively. And we set the embedding and decoder hidden size as 512. T5-based methods are implemented using HuggingFace (Wolf et al., 2020) and inintilized by T5_{base}³. And the hidden size of the GAT layer in the Local Node Encoder is set to 512. For T5-based methods, we set the dropout rate to 0.1, use AdamW optimizer (Loshchilov and Hutter, 2018) and employ a linear learning rate decay schedule without warm-up. We use BLEU (Papineni et al., 2002) for the early stopping criterion. Moreover, the learning rate is 3e-5 and batch size is 4 for all experiments. During decoding, we employ beam search with a beam size 5. All experiments are trained on Nvidia Tesla V100 32GB GPUs.

C.3 Human Evaluation Details

The detailed information about the four human evaluation metrics are as following:

- **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

³<https://huggingface.co/uer/t5-base-chinese-cluecorpussmall>

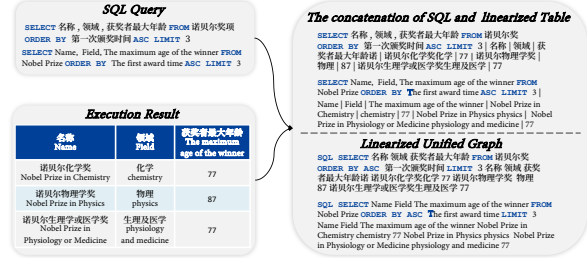


Figure 4: Different linearizations for an input SQL and Table.

and all pieces of information are supported by the table. The score ranges from 1 to 10.

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

For each sample, the annotators need to evaluate four candidates based on the input data. And they do not know which model generates these sentences.

C.4 Differences in COVERAGE between Automatic Evaluation and Human Evaluation

The COVERAGE in Table 4 is calculated by $cov^a = \frac{n^c}{n^a}$, where n^a denotes all cells in the input table and include the cells in the table header. n^c represents the number of cells covered by the generated text. We use string matching rules to determine whether a cell exists in the generated text. cov^a does not consider semantic matching between cells. Therefore, it will miss some cells that are summarized or paraphrased cells.

The COVERAGE in human evaluation is calculated $cov^h = \frac{n^c}{n^t}$, where n^t denotes all cells in the input table and does not include the cells in the table header. n^c represents the number of cells covered by the sentence. n^c is counted by manual checking. Therefore, the cells that are summarized or paraphrased in the generated text will counted.

Overall, cov^a is more rigorous and inflexible than cov^h , and it takes more account of the able headers, so it scores lower.

D Case Study

In Figure 5, we display two decoder output examples from the baselines on the development set of CATS. We find that the model can generate text with high coverage when the input table is simple, such as the number of columns being small. Second, when the input table is complex, such as containing multiple rows and columns, simple models, such as POINTER-GEN, tend to miss some content. Meanwhile, the complex models, such as T5-based ones, only simply enumerate the table cells rather than describe them like humans. Finally, the descriptions generated by models are not faithful to the input, even though they contain most of the input table content. For example, in the second case, all the models do not describe the “earliest” correctly. That is, the descriptions are not logically consistent with the input SQL, which is one of the biggest challenges of this task.

Case A	Input	SQL	SELECT 名称 FROM 八大行星 ORDER BY 最高温度 - 最低温度 ASC LIMIT 3 SELECT name FROM the planets ORDER BY maximum temperature - minimum temperature ASC LIMIT 3
		Table	名称 地球 木星 水星 Name Earth Jupiter Mercury
	Output	GOLD	在八大行星中, 温差最小的3个行星是: 地球、木星、水星。 In the eight planets, the three planets with smallest temperature different are: Earth, Jupiter, Mercury.
		TEMP	在八大行星中, 最高温度 - 最低温度的名称分别是地球、和木星、和水星。 In the eight planets, the names of maximum temperature - minimum temperature are: Earth, Jupiter, and Mercury.
		POINTER-GEN	按温度-温度最大的3个八大行星分别是: 1、地球; 2、木星; 3、水星。 The three eight planets with the largest temperature-temperature are: 1. Earth; 2. Jupiter; 3. Mercury.
		T5-PNN	八大行星的最高温度最高的3个行星分别是: 1、地球; 2、木星; 3、水星。 The three planets with the highest maximum temperatures of the eight planets are: 1. Earth; 2. Jupiter; and 3. Mercury.
OURS	温差最小的3个行星分别是: 1、地球; 2、木星; 3、水星。 The three planets with smallest temperature different are: 1. Earth, 2. Jupiter, 3. Mercury.		
Case B	Input	SQL	SELECT 名称, 领域, 获奖者最大年龄 FROM 诺贝尔奖项 ORDER BY 第一次颁奖时间 ASC LIMIT 3 SELECT name, field, the maximum age of the winner FROM Nobel Prize ORDER BY the first award time DESC LIMIT 3
		Table	名称 领域 获奖者最大年龄 诺贝尔化学奖 化学 77 诺贝尔物理学奖 87 诺贝尔生理学或医学奖 生理及医学 77 name field the maximum age of the winners Nobel Prize in Chemistry chemistry 77 Nobel Prize in Physics physics 87 Nobel Prize in Physiology or Medicine physiology and medicine 77
	Output	Gold	第一次颁奖时间最早的3个诺贝尔奖项的名称、领域、获奖者最大年龄分别是: 1. 化学领域, 获得诺贝尔化学奖的最大年龄是77岁; 2. 物理领域, 获得诺贝尔物理学奖的最大年龄是87岁; 3. 生理及医学领域, 获得诺贝尔生理学或医学奖的最大年龄是77岁。 The fields and maximum ages of the winners of the three Nobel Prizes that were awarded the earliest for the first time are 1. Chemistry, where the maximum age to receive the Nobel Prize in Chemistry is 77; 2. Physics, where the maximum age to receive the Nobel Prize in Physics is 87; 3. Physiology and Medicine, where the maximum age to receive the Nobel Prize in Physiology or Medicine is 77.
		TEMP	按照第一次颁奖的名称、领域、获奖者最大年龄分别是诺贝尔化学奖、化学、77、和诺贝尔物理学奖、物理、87、和诺贝尔生理学或医学奖、生理及医学、77。 According to the name of the first award, the fields and the maximum age of the winners are Nobel Prize in Chemistry, Chemistry, 77, and Nobel Prize in Physics, Physics, 87, and Nobel Prize in Physiology or Medicine, Physiology and Medicine, 77, respectively.
		POINTER-GEN	第一次颁奖时间最少的3个诺贝尔奖项的名称、领域、获奖者最大年龄分别是: 1. 诺贝尔化学奖, 化学, 77; 2. 诺贝尔物理学奖, 物理, 87; 3. 诺贝尔生理学或医学奖, 生理及医学, 77。 The names, fields, and maximum ages of the winners of the three Nobel Prizes that are awarded the minimum for the first time are 1. the Nobel Prize in chemistry, chemistry, 77; 2. Nobel Prize in physics, physics, 87; 3. Nobel Prize in physiology or medicine, physiology, and medicine, 77.
		T5-PNN	第一次颁奖时间最长的3个诺贝尔奖项名称、领域、获奖者最大年龄有3个, 分别是: 1. 诺贝尔化学奖, 化学, 77; 2. 诺贝尔物理学奖, 物理, 87; 3. 诺贝尔生理学或医学奖, 生理及医学, 77。 The fields and maximum ages of the winners of the three Nobel Prizes that are awarded the longest for the first time are 1. the Nobel Prize in Chemistry, Chemistry, 77; 2. the Nobel Prize in Physics, Physics, 87; 3. the Nobel Prize in Physiology or Medicine, Physiology and Medicine, 77.
OURS	第一次颁奖时间最短的3个诺贝尔奖项的名称和获奖者最大年龄分别是: 1. 诺贝尔化学奖, 化学, 77; 2. 诺贝尔物理学奖, 物理, 87; 3. 诺贝尔生理学或医学奖, 生理及医学, 77。 The fields and maximum ages of the winners of the three Nobel Prizes that are awarded the shortest for the first time are 1. the Nobel Prize in Chemistry, Chemistry, 77; 2. the Nobel Prize in Physics, Physics, 87; 3. the Nobel Prize in Physiology or Medicine, Physiology and Medicine, 77.		

Figure 5: Answer-to-sequence examples in CATS. Error words are in red. Confusing and incomprehensible phrases are in blue.