CSS: A Large-scale Cross-schema Chinese Text-to-SQL Medical Dataset

Hanchong Zhang[1*], **Jieyu Li**[1*], **Lu Chen**[1†], **Ruisheng Cao**[1],
Yunyan Zhang[2], **Yu Huang**[2], **Yefeng Zheng**[2] **and Kai Yu**[1†]
[1]X-LANCE Lab, Department of Computer Science and Engineering
MoE Key Lab of Artificial Intelligence, SJTU AI Institute
Shanghai Jiao Tong University, Shanghai, China
[2]Tencent Jarvis Lab, Shenzhen, China
{zhanghanchong,oracion,chenlusz,kai.yu}@sjtu.edu.cn

Abstract

The cross-domain text-to-SQL task aims to build a system that can parse user questions into SQL on complete unseen databases, and the single-domain text-to-SQL task evaluates the performance on identical databases. Both of these setups confront unavoidable difficulties in real-world applications. To this end, we introduce the cross-schema text-to-SQL task, where the databases of evaluation data are different from that in the training data but come from the same domain. Furthermore, we present CSS[1], a large-scale **C**ros**S**-**S**chema Chinese text-to-SQL dataset, to carry on corresponding studies. CSS originally consisted of 4,340 question/SQL pairs across 2 databases. In order to generalize models to different medical systems, we extend CSS and create 19 new databases along with 29,280 corresponding dataset examples. Moreover, CSS is also a large corpus for single-domain Chinese text-to-SQL studies. We present the data collection approach and a series of analyses of the data statistics. To show the potential and usefulness of CSS, benchmarking baselines have been conducted and reported. Our dataset is publicly available at https://huggingface.co/datasets/zhanghanchong/css.

1 Introduction

Given the database, the text-to-SQL task (Zhong et al., 2017; Xu et al., 2017) aims to convert the natural language question into the corresponding SQL to complete complicated querying. As the wild usage of relational database, this task attract great attention and has been widely studied in both academic and industrial communities.

Recently, text-to-SQL researches (Hui et al., 2022; Lin et al., 2020; Qi et al., 2022) mainly focus on building a parser under a cross-domain setup (Yu et al., 2018; Wang et al., 2020b), where the databases of the training set and the evaluation set do not overlap. It aims to construct a universal parser that can automatically adapt different domains to inhibit the problem of data scarcity. However, domain-specific knowledge, especially domain convention, is crucial but difficult to transform across different domains under cross-domain setup. Another line of research focuses on the experiment environment where the training data and the evaluation data are based on the same database, which is known as a single-domain setup. A single-domain text-to-SQL system can parse domain knowledge more easily and also has more wide applications in the real world. However, the problem of data scarcity always comes up when security issues and privacy issues exist. Therefore, both of these setups will face particular difficulties when it comes to the real world.

To this end, we introduce the cross-schema setup in this work. The cross-schema text-to-SQL tasks aim to build a text-to-SQL parser that can automatically adapt different databases from the same domain, which can avoid the aforementioned problems. Actually, the cross-schema text-to-SQL also has broad applications in the real world. For example, all the hospital store the information of patients and medical resources in databases with different structures. Most information categories are identical across these databases, for instance, the patient name and the treatment date. Moreover, domain-specific representations such as medicine names in databases and user questions are also commonly used. In this case, we can build a universal in-domain text-to-SQL parser that can be deployed on the new database from the given domain. Compared with the cross-domain setup, a cross-schema parser will not always confront completely unseen domain knowledge. On the other hand, compared with the single-domain setup, the problem of data scarcity can also be inhibited because

[*] The first two authors contributed equally to this work.

[†] The corresponding authors are Lu Chen and Kai Yu.

[1] Our code is publicly available at https://github.com/X-LANCE/medical-dataset

Findings of the Association for Computational Linguistics: ACL 2023, pages 6970–6983
July 9-14, 2023 ©2023 Association for Computational Linguistics

the data from other in-domain databases can be used to train the model. However, a cross-schema text-to-SQL parser need to automatically adapt different database schema structure. Unfortunately, this issue is less investigated before. Therefore, how to construct a structural-general parser is the mainly challenge of cross-domain text-to-SQL.

In this paper, we propose a large-scale **C**rosS-**S**chema Chinese text-to-SQL dataset (CSS), containing 33,620 question/SQL pairs across 21 databases. We generate (question, SQL) pairs with templates and manually paraphrase the question by crowd-sourced. For the databases, we collect 2 real-world database schemas involving medical insurance and medical treatment. As the privacy issues, we are not allowed to use the original data. Therefore, we fill the databases with pseudo values. Based on these 2 seed databases, we alter the schema and expand 19 databases with different structures. Hence, CSS can be used to develop cross-schema text-to-SQL systems. On the other hand, the original 2 databases correspond 4,340 samples, which construct the largest Chinese single-domain corpus. This corpus also allows researchers to carry on related studies. Our main contributions can be summarized as follows:

1. We present the cross-schema text-to-SQL task and propose a large-scale dataset, CSS, for corresponding studies. The dataset and baseline models will be available if accepted.

2. We provide a real-world Chinese corpus for single-domain text-to-SQL researches.

3. To show the potential and usefulness of CSS, we conducted and reported the baselines of cross-schema text-to-SQL and Chinese single-domain text-to-SQL.

2 Related Works

Single-domain text-to-SQL datasets Earliest semantic parsing models are designed for single-domain systems to answer complex questions. ATIS (Price, 1990; Dahl et al., 1994) contains manually annotated questions for the flight-booking task. GeoQuery (Zelle and Mooney, 1996) contains manually annotated questions about US geography. Popescu et al. (2003); Giordani and Moschitti (2012); Iyer et al. (2017) convert GeoQuery into the SQL version. Restaurants (Tang and Mooney, 2000; Popescu et al., 2003) is a dataset including questions about restaurants and their food types etc. Scholar (Iyer et al., 2017) includes questions about academic publications and corresponding automatically generated SQL queries. Academic (Li and Jagadish, 2014) enumerates all query logics supported by the Microsoft Academic Search (MAS) website and writes corresponding question utterances. Yelp and IMDB (Yaghmazadeh et al., 2017) consists of questions about the Yelp website and the Internet Movie Database. Advising (Finegan-Dollak et al., 2018) consists of questions about the course information database at the University of Michigan along with artificial data records.

Single-domain text-to-SQL datasets contain only one database. Although text-to-SQL models trained with single-domain datasets are applied in corresponding specific domains, different systems with the same domain but different backgrounds have diverse databases, which means that models should have the generalization ability to be transferred among different systems. Existing single-domain datasets do not own the feature that requires models to improve cross-schema generalization ability. On the contrary, our cross-schema setup is raised for this issue.

Cross-domain text-to-SQL datasets Recent researches expect text-to-SQL models (Guo et al., 2019; Bogin et al., 2019; Zhang et al., 2019) to generalize to unseen databases. Thus cross-domain text-to-SQL datasets are released. Zhong et al. (2017) releases WikiSQL, a dataset of 80,654 manually annotated question/SQL pairs distributed across more than 20k tables from Wikipedia. Although WikiSQL is a large-scale dataset, each database schema merely consists of one table and each SQL query merely consists of SELECT, FROM, WHERE clauses. Yu et al. (2018) releases Spider, a large-scale complex cross-domain text-to-SQL dataset. Comparing with previous datasets, Spider owns much more complex databases for various domains and complex SQL queries with advanced SQL clauses and nested SQL structures. Wang et al. (2020b) releases DuSQL, yet another large-scale cross-domain text-to-SQL dataset but in Chinese. Having similar form with Spider, DuSQL has become a popular Chinese text-to-SQL dataset. There are also some conversational cross-domain text-to-SQL datasets, including SParC (Yu et al., 2019b), CoSQL (Yu et al., 2019a), CHASE (Guo et al., 2021), DIR (Li et al., 2023b) etc.

Although our cross-schema dataset owns more than one databases, it is different from cross-domain datasets. It concentrates on model generalization ability across different databases which share the similar structure since they are in the same domain.

3 Dataset Collection

In this section, we introduce our method of constructing the medical dataset CSS in detail. The dataset construction method mainly consists of five steps: 1) initial databases creation, 2) question/SQL templates creation, 3) values filling, 4) questions rewriting, and 5) database schema extension.

We discuss five steps of constructing the dataset in Section 3.1-3.5 respectively. Figure 1 shows the overview of the complete process.

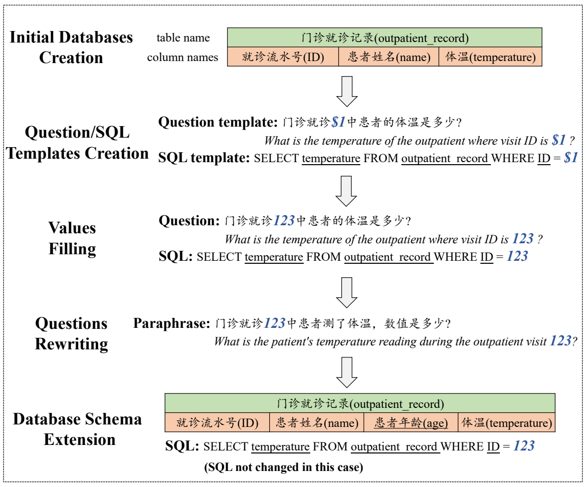

Initial Databases Creation
table name
column names
门诊就诊记录(outpatient_record)
就诊流水号(ID)
患者姓名(name)
体温(temperature)
Question/SQL Templates Creation
Question template: 门诊就诊$1中患者的体温是多少?
What is the temperature of the outpatient where visit ID is $1 ?
SQL template: SELECT temperature FROM outpatient_record WHERE ID = $1
Values Filling
Question: 门诊就诊123中患者的体温是多少?
What is the temperature of the outpatient where visit ID is 123 ?
SQL: SELECT temperature FROM outpatient_record WHERE ID = 123
Questions Rewriting
Paraphrase: 门诊就诊123中患者测了体温，数值是多少?
What is the patient's temperature reading during the outpatient visit 123?
Database Schema Extension
门诊就诊记录(outpatient_record)
就诊流水号(ID)
患者姓名(name)
患者年龄(age)
体温(temperature)
SQL: SELECT temperature FROM outpatient_record WHERE ID = 123
(SQL not changed in this case)

Figure 1: Overview of the dataset collection process.

3.1 Initial Databases Creation

To construct the dataset, the first step is to create initial databases. We collect two databases from the real world scenario, i.e. the insurance database and the medical database. The insurance database mainly stores medical consumption records of many different patients. The medical database mainly stores records of medical diagnostic and examination results.

It is obvious that records data in medical databases are usually sensitive, since the issue of patients privacy is involved in these data. It is not feasible to use data from the real world directly in our dataset. To protect privacy of users involved in the medical system, we generate database cell-values with certain rules and ensure that generated data are reasonable.

3.2 Question/SQL Templates Creation

Creating abundant and diverse question/SQL templates is an important step for constructing the dataset, which influences the quality of the generated dataset a lot. A question/SQL template can be regarded as an example of the dataset, which consists of a question template and a SQL query template answering the question. The only difference between the question/SQL template and the real dataset example is that values carrying information (e.g. ID, name, time) in the question/SQL template are replaced with special tokens. In the subsequent steps, values can be generated and filled into corresponding question/SQL templates with certain rules, which means that all question/SQL templates can be transformed into real dataset examples eventually.

In general, we use three methods to create various question/SQL templates. Firstly, given medical databases, we enumerate all columns and attempt to raise a question for each column as far as possible. Sometimes we put several columns with close lexical relations into one question/SQL template, since the diversity of the SELECT clause can get increased. It is obvious that question/SQL templates written by this method are relatively simple.

Secondly, we raise a few medical query scenarios and create question/SQL templates based on them. In the real world, different people with different occupations and social roles will ask different types of questions. For instance, patients may care their medical consumption records and doctors may care medical examination results. Based on different real-world scenarios, we can raise various questions that meet needs of people with different social roles (e.g. doctor, patient). Furthermore, these question/SQL templates are usually more challenge since their SQL skeletons are usually more complex and diverse.

Thirdly, we add question/SQL templates which include SQL keywords and SQL skeletons that never occur in previous templates. We count occurrence frequencies for all SQL grammar rules and SQL skeletons that occur in dataset examples. Referring to statistical results, we create questions and corresponding SQL queries which consist of SQL grammar rules that occur in few dataset examples. Detailed statistical results are shown in Section 4.2. By creating question/SQL templates with this method, the SQL diversity of the dataset can get improved.

We eventually raise 434 different question/SQL templates totally. All these templates will get processed in subsequent steps.

3.3 Values Filling

In order to generate real dataset examples from question/SQL templates, values should be generated and filled into all templates. Different types of values are replaced with different special tokens in question/SQL templates. In this step, we use certain rules to generate random values for various special tokens.

Concretely, special tokens indicating number or time are filled with reasonable and suitable random values. Special tokens indicating ID (e.g. person ID, hospital ID) are filled with random strings, which consist of numbers and letters. Other special tokens basically indicate specialized and professional words like disease names. To generate these values, we firstly collect sufficient disease names, medicine names, medical test names, etc. Then these special tokens are filled with values chosen at random from corresponding candidate value lists.

Actually one unique question/SQL template can be used to generate several different dataset examples, since the template can be completed with various random values. We generate 10 dataset examples for each question/SQL template. Consequently there are totally 4,340 question/SQL pairs which are directly generated from 434 question/SQL templates.

3.4 Questions Rewriting

Although 4,340 question/SQL pairs directly generated from templates can already be used to train and test text-to-SQL models, they cannot be directly added into the eventual medical dataset. Question sentences generated from question templates are usually unnatural. Moreover, 10 question sentences generated from the same one question template share the same sentence pattern. which means lack of natural language diversity.

To tackle the issue of language naturalness and diversity, we recruit annotators to rewrite dataset examples. All questions directly derived from question templates are rewritten by annotators. In this process, lexical and syntactic patterns of question sentences get changed, which leads to improvement of natural language diversity of the dataset.

To ensure the diversity of rewritten question sentences, we design a specific metric to evaluate the rewriting quality. We recruit two groups of annotators and request them to rewrite question sentences with metric scores as high as possible. Finally we merge two rewriting results from different annotating groups with some rules and acquire all rewritten questions. Detailed explanation of the metric is shown in Appendix A.

The correctness of rewritten questions is also an important issue. We use the automatic method to examine rewritten questions and make sure that key information are always maintained after the rewriting process.

Payment. All annotators were paid based on their annotations. Annotators would get paid 0.58 RMB for each annotation example.

3.5 Database Schema Extension

Database schema extension is a key feature of CSS. Text-to-SQL models with good performance should have the ability to be used in various medical systems. In the real world application, different medical systems may use different databases. However, these databases may share the similar structure, since all of them are designed for the medical domain. Consequently, we believe that cross-schema generalization ability for text-to-SQL models is significant and add this challenge task in CSS.

CSS originally contains 2 databases. Based on them, we follow Li et al. (2023a) and create 19 new databases. Firstly for two tables linked with foreign keys, we create a new relation table between the original two tables and create new foreign keys respectively pointing to them. Secondly for two tables linked with foreign keys, we merge them by putting their columns together in a merged table. Thirdly for a table with a special column which only contains a few different kinds of values (e.g. gender), we split the table into several tables according to those limited values.

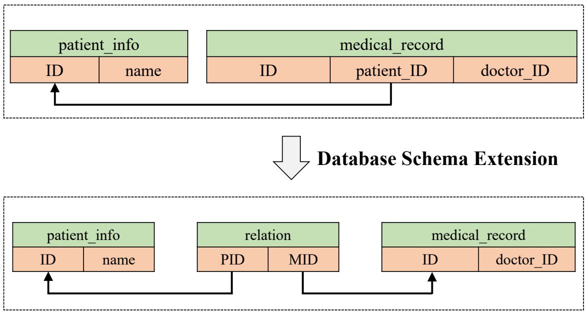


Figure 2: An instance of database schema extension.

After creating databases, CSS acquires 19 new databases and 29,280 new dataset examples. Therefore, CSS totally contains 33,620 question/SQL pairs across 21 databases.

4 Dataset Statistics and Comparison

In this section, we list some statistical information of CSS and existing datasets and do comparison. We mainly discuss scale statistics and SQL statistics with various datasets, including single-domain datasets, cross-domain datasets and CSS.

4.1 Scale Statistics

Table 1 shows scale statistics of existing datasets, including single-domain datasets, cross-domain datasets, and the medical dataset CSS. For single-domain datasets listed in the table and WikiSQL, we use the standardized version from Finegan-Dollak et al. (2018). CSS contains 33,620 examples generated from scratch across 21 databases. Comparing with previous single-domain datasets, CSS has the largest scale and various databases. We extend original databases with several certain rules. Therefore, CSS can help text-to-SQL models generalize to different medical systems, where databases are different but share the similar structure.

Databases in CSS have a great number of columns, composite primary keys, and foreign keys, which indicates that databases in CSS commonly possess complex structures. This is also a challenge feature of CSS. It requires models to find out effective information from complex database structures.

4.2 SQL Statistics

First of all, we clarify the concept named SQL skeleton. For a certain SQL query, it is feasible to remove detailed schema items and values from the SQL query. Concretely, we replace tables used in the SQL query with the special token "tab". Columns and values are processed with the similar method. Columns are replaced with the special token "col" and values are replaced with the special token "value". Then the result is defined as the SQL skeleton, which retains the basic structure of the original SQL query.

Table 2 shows SQL statistics of existing datasets. CSS totally possesses 562 different SQL skeletons, which is comparable with ATIS and surpasses other single-domain datasets. Note that SQL queries in CSS are commonly very long. The average and maximum number of SQL query tokens are 55.41 and 243 respectively, which has surpassed almost all existing datasets except ATIS. The statistical result indicates that SQL queries in CSS are diverse and complex. This is still a challenge for text-to-SQL models.

5 Tasks and Models

5.1 Dataset Splitting

We provide three methods to split the dataset into train/dev/test sets. Different dataset splitting methods correspond to different tasks and raise different challenges for models. For the first method, 4,340 original dataset examples are shuffled at random and then are split with the ratio 0.8/0.1/0.1. This sub-task is an ordinary text-to-SQL task setting and requires models to generalize well on natural language.

For the second method, 434 question/SQL templates are shuffled at random and then are split with the ratio 0.8/0.1/0.1. Then 4,340 original question/SQL pairs fall into corresponding dataset subsets. Comparing with other dataset splitting methods, larger language gap and SQL gap exist among train/dev/test sets, since different question/SQL templates generally express different meanings. Models are required to have the stronger SQL pattern generalization ability under this sub-task.

For the third method, we add extended dataset examples and split all 33,620 examples according to their databases. All databases are split with the ratio 0.6/0.2/0.2. No overlap of databases exists in train/dev/test sets. This dataset splitting method provides a challenge task, which requires models to possess the stronger generalization ability across diverse databases sharing similar structures.

5.2 Syntactic Role Prediction

How to improve the cross-schema generalization ability of text-to-SQL models is a key challenge raised in CSS. In this section, we introduce our simple method to tackle the issue of model generalization ability across different databases.

The text-to-SQL model LGESQL (Cao et al., 2021) add an auxiliary task named graph pruning in order to improve the model performance. Given the natural language question and the database schema, the model is required to predict whether each schema item occurs in the SQL query. Following Cao et al. (2021), we raise a similar auxiliary task named syntactic role prediction (SRP). Under

Dataset	Language	Examples	DBs	Avg T/DB	Avg C/T	Avg P/T	Avg F/T
ATIS	English	19,201	1	25	5.24	0.16	1.56
GeoQuery	English	920	1	8	3.88	1.75	1.12
Restaurants	English	378	1	3	4.00	1.00	1.33
Scholar	English	1,858	1	12	2.33	0.58	0.75
Academic	English	200	1	15	2.80	0.47	0.00
Yelp	English	141	1	7	5.43	1.00	0.00
IMDB	English	147	1	16	4.06	1.00	0.19
Advising	English	4,744	1	18	6.89	1.39	5.39
WikiSQL	English	80,654	26,531	1.00	6.34	0.00	0.00
Spider	English	9,693	166	5.28	5.14	0.89	0.91
DuSQL	Chinese	25,003	208	4.04	5.29	0.51	0.71
CSS	Chinese	33,620	21	5.62	28.49	1.68	1.65

Table 1: Scale statistics of existing datasets. "Avg T/DB" represents the average number of tables per database schema. "Avg C/T" represents the average number of columns per table. "Avg P/T" represents the average number of columns in the composite primary key per table. "Avg F/T" represents the average number of foreign keys per table.

Dataset	# SQL	Avg Len	Max Len
ATIS	828	97.96	474
GeoQuery	120	26.08	92
Restaurants	12	29.22	61
Scholar	158	37.07	65
Academic	76	36.30	116
Yelp	62	28.92	56
IMDB	30	27.48	55
Advising	169	47.49	169
WikiSQL	39	12.48	23
Spider	1,116	17.99	87
DuSQL	2,323	20.23	37
CSS	562	55.41	243

Table 2: SQL statistics of existing datasets. "# SQL" represents the number of SQL skeletons. "Avg Len" represents the average number of tokens in one SQL query. "Max Len" represents the maximum number of tokens in one SQL query.

this task, the model is required to predict in which SQL clause each question token occurs.

The SQL query structure may change as the database schema changes. Figure 3 shows an instance, where two databases share the similar structure but the key information "doctor" in the question are used in the FROM clause and the WHERE clause respectively. We hypothesize that model with strong cross-schema generalization ability should distinguish syntactic roles of every question tokens under different databases.

Concretely, according to the text-to-SQL model LGESQL, the model input is a graph $G = (V, E)$ constructed with the given question and the database schema. Graph nodes V include question tokens and schema items (i.e. tables and columns) and graph edges E indicate relations among them. The model encodes each node i into an embedding vector $\mathbf{x}_i$. Then the context vector $\tilde{\mathbf{x}}_i$ for each node i can be computed with multi-head attention.

$$\alpha_{ij}^h = \text{softmax}_{j \in \mathcal{N}_i} \frac{(\mathbf{x}_i \mathbf{W}_q^h)(\mathbf{x}_j \mathbf{W}_k^h)^{\mathrm{T}}}{\sqrt{d/H}},$$
$$\tilde{\mathbf{x}}_i = (\text{concat}_{h=1}^H \sum_{j \in \mathcal{N}_i} \alpha_{ij}^h \mathbf{x}_j \mathbf{W}_v^h) \mathbf{W}_o,$$

where d is the dimension of embedding vectors, H is the number of heads, $\mathcal{N}_i$ is the neighborhood of the node i, and $\mathbf{W}_q^h, \mathbf{W}_k^h, \mathbf{W}_v^h \in \mathbb{R}^{d \times d/H}$, $\mathbf{W}_o \in \mathbb{R}^{d \times d}$ are network parameters.

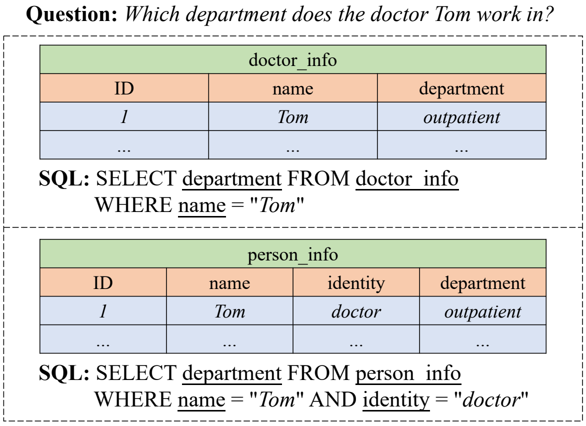
For each question node q_i, the model can predict in which SQL clause it occurs with $\mathbf{x}_{q_i}$ and $\tilde{\mathbf{x}}_{q_i}$. Specifically we divide the SQL query into 16 different parts, which are discussed in detail in Appendix B. Thus the auxiliary task is a binary classification task for each question token and each SQL part.

$$P(\mathbf{y}_{q_i} | \mathbf{x}_{q_i}, \tilde{\mathbf{x}}_{q_i}) = \sigma([\mathbf{x}_{q_i}; \tilde{\mathbf{x}}_{q_i}] \mathbf{W} + \mathbf{b}),$$

where $W \in \mathbb{R}^{2d \times 16}, b \in \mathbb{R}^{1 \times 16}$ are network parameters and $\mathbf{y}_{q_i}$ is the probability vector. The ground truth $y_{q_i,j}^g$ is 1 when the question token q_i occurs in the j-th SQL part. The training object is

$$\mathcal{L} = -\sum_{q_i} \sum_{j} [y_{q_i,j}^g \log P(y_{q_i,j} | \mathbf{x}_{q_i}, \tilde{\mathbf{x}}_{q_i}) + (1 - y_{q_i,j}^g) \log(1 - P(y_{q_i,j} | \mathbf{x}_{q_i}, \tilde{\mathbf{x}}_{q_i})].$$

The syntactic role prediction task is combined with the main task in a multitasking way. In addition, SRP can also be added into the RATSQL model directly, since RATSQL and LGESQL both encode graph nodes into embedding vectors and SRP only takes these vectors as the input.



Question: *Which department does the doctor Tom work in?*

doctor_info		
ID	name	department
1	*Tom*	*outpatient*
...	...	...

SQL: SELECT department FROM doctor_info
WHERE name = "*Tom*"

person_info			
ID	name	identity	department
1	*Tom*	*doctor*	*outpatient*
...	...	...	...

SQL: SELECT department FROM person_info
WHERE name = "*Tom*" AND identity = "*doctor*"

Figure 3: Given the question, the corresponding SQL query differs among various but similar databases.

6 Experiments

6.1 Experiment Setup

Baseline approaches We adopt three competitive text-to-SQL models as the baseline approaches, i.e. RATSQL (Wang et al., 2020a), LGESQL (Cao et al., 2021), and PICARD (Scholak et al., 2021). RATSQL and LGESQL process given information with graph encoding and decode the abstract syntax tree (AST) of the result SQL query. PICARD is a sequence-to-sequence approach and is different from the other two approaches.

RATSQL constructs a graph with question tokens and schema items (i.e. tables and columns) and encodes the graph with the relation-aware self-attention mechanism. With the unified framework, RATSQL can easily establish and handle relations among graph nodes and then encode elements with various categories jointly.

Comparing with RATSQL, LGESQL improves the model performance by utilizing the line graph. LGESQL pays more attention to the topological structure of graph edges and distinguishes local and non-local relations for graph nodes. Besides the original graph used in RATSQL, LGESQL also constructs the corresponding line graph, since the line graph can help facilitate propagating encoding messages among nodes and edges.

Model	**Dev**		**Test**	
	w/o	**w**	**w/o**	**w**
RATSQL	90.2	81.1	89.0	79.1
LGESQL	91.7	82.2	90.8	81.1
PICARD	93.8	53.7	70.3	58.3

Table 3: Model performances under dataset splitting method according to examples.

Different from RATSQL and LGESQL, PICARD is a sequence-to-sequence model. Nowadays large pretrained language models have possessed the strong ability for handling and processing natural language with unconstrained output space. However, SQL is a formal language with strict grammar rules. Invalid SQL queries are very likely to be generated if pretrained models are directly finetuned with text-to-SQL datasets. PICARD provides an approach, which can help reject invalid tokens during each decoding step and generate sequences in the constrained output space.

For each baseline model, we use pretrained language models (PLMs) within the encoding module. In our experiments, the PLM `longformer-chinese-base-4096` is applied in RATSQL and LGESQL and the PLM `mbart-large-50` is applied in PICARD.

Evaluation metrics There are several metrics to evaluate text-to-SQL model performances, including exact match and execution accuracy etc. The exact match metric requires the predicted SQL query to be equivalent to the gold SQL query. The execution accuracy metric requires the execution result of the predicted SQL query to be correct.

We mainly use the exact match (EM) metric in our experiments. Concretely, we present model performances with (w) and without (w/o) value evaluation respectively.

6.2 Results and Analysis

According to 3 different dataset splitting methods, we test baseline models under 3 sub-task settings. Table 3 shows model performances under dataset splitting method according to examples. LGESQL achieves the best performance under this sub-task, i.e. 90.8% EM(w/o) accuracy and 81.1% EM(w) accuracy on the test set. This indicates that existing text-to-SQL parsing models have had the ability to perform very well if all databases and possible SQL structures have appeared in the train set. Models merely need to generalize on natural language, which is simple when utilizing strong PLMs.

Table 5 shows model performances under the template-splitting sub-task. Comparing with the previous sub-task, performances of three baseline models decrease a lot. Although RATSQL achieves the best performance under this sub-task, the EM(w/o) accuracy and the EM(w) accuracy on the test set are only 58.9% and 53.0% respectively. Question/SQL templates in dev/test sets do not appear in the train set. Thus models have to predict unseen SQL patterns when testing. The experiment result indicates that there is still a large room for the improvement of model generalization ability across SQL patterns. We believe that CSS can also help facilitate researches on improving model ability of predicting unseen SQL patterns.

Model	**Dev**		**Test**	
	w/o	**w**	**w/o**	**w**
RATSQL	60.5	55.2	58.9	53.0
LGESQL	59.5	54.4	58.5	52.8
PICARD	52.6	40.9	49.8	38.6

Table 5: Model performances under dataset splitting method according to templates.

Model	**Dev**		**Test**	
	w/o	**w**	**w/o**	**w**
RATSQL	36.6	35.7	43.4	42.0
RATSQL + SRP	38.3	37.4	47.2	45.3

Table 6: Model performances under dataset splitting method according to databases. "SRP" represents the auxiliary task named syntactic role prediction.

Q: 列出水天干这位患者在医院7539997住院的就诊记录里入院科室名字含有耳鼻喉科的记录

Q: *List the records of patient Tiangan Shui admitted to hospital 7539997, including the records with the department name containing Otolaryngology.*

Gold: SELECT * FROM person_info JOIN hz_info JOIN zyjzjlb WHERE person_info.XM = "水天干" AND hz_info.YLJGDM = "*7539997*" AND zyjzjlb.JZKSMC LIKE "%耳鼻喉科%"

Pred: SELECT * FROM person_info JOIN hz_info JOIN zyjzjlb WHERE person_info.XM = "水天干" AND hz_info.YLJGDM = "*7539997*" AND zyjzjlb.JZKSMC LIKE "%耳鼻炎%"

Q: 从01年1月31日一直到09年8月12日内患者80476579被开出盐酸多奈哌齐片(薄膜)的总次数一共有多少?

Q: *How many times has patient 80476579 been prescribed donepezil hydrochloride tablets (thin film) from 2001-01-31 to 2009-08-12?*

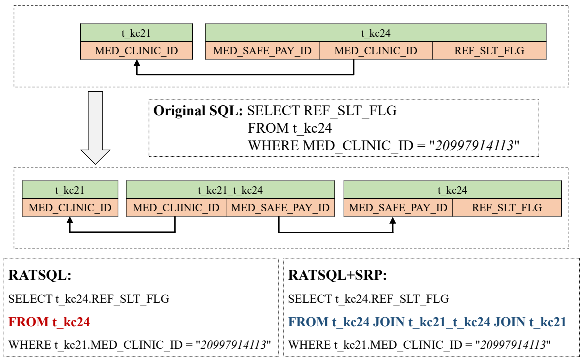
Gold: SELECT COUNT(*) FROM t_kc21 JOIN t_kc22 WHERE t_kc21.PERSON_ID == "*80476579*" AND t_kc22.STA_DATE BETWEEN "*2001-01-31*" AND "*2009-08-12*" AND t_kc22.SOC_SRT_DIRE_NM == "盐酸多奈哌齐片(薄膜)"

Pred: SELECT COUNT(*) FROM t_kc21 JOIN t_kc22 WHERE t_kc21.PERSON_ID == "*80476579*" AND t_kc22.STA_DATE BETWEEN "*2001-01-31*" AND "*2009-08-12*" AND t_kc22.SOC_SRT_DIRE_NM == "盐酸多奈"

Table 4: Case study for the PICARD model when predicting values. FROM conditions are omitted for clarity.

Note that as a sequence-to-sequence approach, PICARD cannot perform as well as the two AST-based approaches (RATSQL and LGESQL) in the template-splitting sub-task. There is a room of model performances between PICARD and AST-based approaches, especially when values in SQL queries are concerned in evaluation. Table 4 shows two instances from the test set in the template-splitting sub-task, where the PICARD model successfully generates the structure of the SQL query but predicts the wrong value. As shown in Table 2, SQL queries in CSS are commonly very long and complex, which leads to great difficulty for PICARD decoding. The decoding error would accumulate as the decoding step increases. According to our statistical results, during the decoding process of AST-based approaches, the average number of AST nodes is 56.95. Although the average number of tokens in the SQL query is 55.41, PLM used in PICARD would split tokens into many subwords. Consequently, decoding steps of PICARD is actually much more than AST-based approaches. Furthermore, table and column names in CSS are commonly consisted of unnatural tokens, which improves the decoding difficulty of PICARD a lot.

Table 6 shows model performances under dataset splitting method according to different databases. Under this sub-task, we use RATSQL as the baseline model and attempt to add the auxiliary task SRP, expecting to improve the model performance across different databases. The experiment result shows that the model performance increases about 1.7% on the dev set and increases about 3.3%-3.8% on the test set when SRP is applied into RATSQL. This proves that SRP can help improve the cross-schema generalization ability of the model when using SRP as a simple baseline method.



Original SQL: SELECT REF_SLT_FLG
FROM t_kc24
WHERE MED_CLINIC_ID = "*20997914113*"

RATSQL:
SELECT t_kc24.REF_SLT_FLG
FROM t_kc24
WHERE t_kc21.MED_CLINIC_ID = "*20997914113*"

RATSQL+SRP:
SELECT t_kc24.REF_SLT_FLG
FROM t_kc24 JOIN t_kc21_t_kc24 JOIN t_kc21
WHERE t_kc21.MED_CLINIC_ID = "*20997914113*"

Figure 4: Case study for SRP. JOIN conditions in the FROM clause are omitted for brevity.

Figure 4 is an instance from the test set, where RATSQL predicts the wrong SQL but RATSQL with SRP predicts the correct result. After database schema extension, a new relation table is created. However, RATSQL does not understand the change and misses the relation table in the FROM clause. On the contrary, the auxiliary task SRP helps the model utilize the relation table and eventually predict the correct SQL.

7 Conclusion

This paper presents CSS, a large-scale cross-schema Chinese text-to-SQL dataset designed for the medical domain. We illustrate the detailed process of dataset construction and also present statistical information comparing with existing datasets. We raise a challenge task in CSS, which requires models to generalize across various databases but in the same domain. To tackle the above task, we designed a baseline method named syntactic role prediction as an auxiliary task for model training. We conduct benchmark experiments with three competitive baseline models and prove that future researches on CSS is valuable.

Limitations

We raise a new challenge task in our medical dataset CCS. Comparing with existing datasets, CCS requires text-to-SQL models to generalize to different databases with the similar structure in the same domain. To tackle this problem, we provide a baseline method named syntactic role prediction, which is an auxiliary task and can be combined with the main task in a multitasking way. Our experiments prove that SRP can help improve the cross-schema generalization ability of models. However, the improvement is not that large. How to generalize models across different databases sharing the similar structure is still a challenge issue. We expect that future works can solve this difficult problem.

Ethics Statement

We collect two original medical databases from the real world. However, cell-values in medical databases are commonly sensitive, since the information of patients and doctors are involved in these values. Thus we only retain the database schema and generate sufficient cell-values with certain rules. We ensure that generated values are reasonable and that privacy of medical system users can get protected.

Acknowledgments

We thank Xiaowen Li, Kunrui Zhu and Ruihui Zhao from Tencent Jarvis Lab for providing necessary initial data. We also thank all the anonymous reviewers for their thoughtful comments. This work has been supported by the China NSFC Project (No.62106142 and No.62120106006), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), CCF-Tencent Open Fund and Startup Fund for Youngman Research at SJTU (SFYR at SJTU).

References

Ben Bogin, Jonathan Berant, and Matt Gardner. 2019. Representing schema structure with graph neural networks for text-to-SQL parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4560–4565, Florence, Italy. Association for Computational Linguistics.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. LGESQL: Line graph enhanced text-to-SQL model with mixed local and non-local relations. 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 2541–2555, Online. Association for Computational Linguistics.

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

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, Melbourne, Australia. Association for Computational Linguistics.

Alessandra Giordani and Alessandro Moschitti. 2012. Translating questions to SQL queries with generative parsers discriminatively reranked. In *Proceedings of COLING 2012: Posters*, pages 401–410, Mumbai, India. The COLING 2012 Organizing Committee.

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

Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Yanyang Li, Bowen Li, Jian Sun, and Yongbin Li. 2022. S^{2}SQL: Injecting syntax to question-schema interaction graph encoder for text-to-SQL parsers. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1254–1262, Dublin, Ireland. Association for Computational Linguistics.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada. Association for Computational Linguistics.

Fei Li and H. V. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proc. VLDB Endow.*, 8(1):73–84.

Jieyu Li, Lu Chen, Ruisheng Cao, Su Zhu, Hongshen Xu, Zhi Chen, Hanchong Zhang, and Kai Yu. 2023a. On the structural generalization in text-to-sql.

Jieyu Li, Zhi Chen, Lu Chen, Zichen Zhu, Hanqi Li, Ruisheng Cao, and Kai Yu. 2023b. Dir: A large-scale dialogue rewrite dataset for cross-domain conversational text-to-sql. *Applied Sciences*, 13(4).

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4870–4888, Online. Association for Computational Linguistics.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, page 149–157, New York, NY, USA. Association for Computing Machinery.

P. J. Price. 1990. Evaluation of spoken language systems: the ATIS domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990.*

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. RASAT: Integrating relational structures into pretrained Seq2Seq model for text-to-SQL. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3215–3229, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

Lappoon R. Tang and Raymond J. Mooney. 2000. Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 133–141, Hong Kong, China. Association for Computational Linguistics.

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, Online. 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)*, pages 6923–6935, Online. Association for Computational Linguistics.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Type- and content-driven synthesis of SQL queries from natural language. *CoRR*, abs/1702.01168.

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 Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 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, Hong Kong, China. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019b. SParC: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523, Florence, Italy. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, page 1050–1055. AAAI Press.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-based SQL query generation for cross-domain context-dependent questions. 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 5338–5349, Hong Kong, China. Association for Computational Linguistics.

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

A Rewriting Metric

First of all, we define the rewriting ratio (RR) between two different sentences s_1 and s_2, i.e.

$$RR(s_1, s_2) = \frac{\mathrm{EditDistance}(s_1, s_2)}{|s_1| + |s_2|},$$

where $\mathrm{EditDistance}(s_1, s_2)$ represents the edit distance between s_1 and s_2. Assume that $s_{i,1}, s_{i,2}, \cdots, s_{i,10}$ are ten rewritten question sentences derived from the same question/SQL template i. In order to improve the language diversity, we expect ten rewritten sentences to differ from each other. Thus we request annotators to maximize

$$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{55}\sum_{1\le j<k\le 10} RR(s_{i,j}, s_{i,k}),$$

when rewriting, where N is the number of question/SQL templates.

When merging rewriting results from two groups of annotators, for each example with the original question sentence s^o, we need to decide between two rewritten sentences s_1^r and s_2^r. Here we choose s_1^r only if

$$RR(s^o, s_1^r) > RR(s^o, s_2^r).$$

B Syntactic Role Prediction

We divide the SQL query into 16 different parts. Table 7 shows detailed situations. For each question token q_i, we find out all schema items which have schema linking relations with q_i. Then for each SQL part, we label that q_i appears in this part if q_i itself or one of those schema items appears in this part.

Name	**Description**
NONE	Element is not used in SQL.
SELECT	Element is a normal column in SELECT.
SELECT_AGG	Element is a column with an aggregation function in SELECT.
SELECT_NEST	Element appears in SELECT, where SELECT is a nested SQL query.
FROM	Element is a normal table in FROM.
FROM_NEST	Element appears in FROM, where FROM is a nested SQL query.
WHERE	Element is a normal column in WHERE
WHERE_NEST	Element appears in WHERE, where WHERE is a nested SQL query
GROUP	Element is a normal column in GROUP BY.
HAVING	Element appears in HAVING.
ORDER	Element is a normal column in ORDER BY.
ORDER_AGG	Element is a column with an aggregation funciton in ORDER BY.
LIMIT	Element appears in LIMIT.
INTERSECT	Element appears in INTERSECT.
UNION	Element appears in UNION.
EXCEPT	Element appears in EXCEPT.

Table 7: 16 parts of the SQL query.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
8

☑ A2. Did you discuss any potential risks of your work?
8

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

5,6

☑ B1. Did you cite the creators of artifacts you used?
5,6

□ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

□ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
3

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
3

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
4

C ☑ Did you run computational experiments?

5,6

☒ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
It is not important.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☒ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
It is not important.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
5,6

□ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Not applicable. Left blank.

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?
3

☑ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
3

☑ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
3

☑ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
3,5,6

☑ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
3

☑ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
3