Crit2SQL: Dataset and Analysis for Criteria2SQL Generation

Anonymous ACL submission

Abstract

Converting clinical trial criteria to Structured Query Language (Criteria2SQL) is crucial for efficient patient recruitment in clinical studies. As manual conversion could be onerous, large-scale pre-trained language models have recently be exploited to automatically generate SQL based on the input text. To further promote research in Criteria2SQL, we propose a large-scale manually-annotated dataset Crit2SQL, which annotates SQL queries for 4,000 real-world clinical trial criteria obtained from ClinicalTrials.gov. We conduct an in-depth analysis of Codex's generation on this dataset by summarizing its errors into seven types. Our analysis reveals the limitation of applying general Text2SQL model to Criteria2SQL, necessitating the development of domain-specific approaches. Compared to general Text2SQL tasks, criteria in Criteria2SQL task are categorized as inclusion and exclusion criteria. By leveraging such information to construct positive and negative pairs, we propose a novel contrastive-learning-based Criteria2SQL generation approach, which achieve at least 11.72% accuracy improvements compared to two existing Criteria2SQL methods. Our dataset, analysis, and method collectively serve as the basis for building accurate Criteria2SQL models, paving the path for efficient clinical trial recruitment.

1 Introduction

Clinical trials are essential for studying new drugs and treatments, which serve as the basis for precision therapy and drug discovery (Zarin et al., 2011; Tse et al., 2009; Califf et al., 2012). One of the most important factors for a successful clinical trials is recruiting patients through a few pre-defined criteria (Lovato et al., 1997). These criteria are written in plain text by clinician who designed the trial (Liu et al., 2021). To apply these criteria to patient selection in a patient database, we need to convert them to Structured Query Language (SQL) (Yu et al., 2020; Wang et al., 2020). This conversion could be onerous and challenging for clinician (Pan et al., 2021; Choi et al., 2021; Lee et al.). Therefore, there is a pressing need to develop an automated approach to convert clinical trial criteria to SQL (Criteria2SQL).

Recently, large-scale pre-trained language models (PLMs) have demonstrated promising performance in the task of Text2SQL (Wang et al., 2020; Pan et al., 2021; Chen et al., 2021; Guo et al., 2019; Yu et al., 2020; Rajkumar et al., 2022; Wang et al., 2022; Dou et al., 2022; Katsogiannis-Meimarakis and Koutrika, 2021). Criteria2SQL can be regarded as a domain-specific Text2SQL task, where the text are short clinical trial criteria. However, in contrast to the text in other Text2SQL tasks, clinical trial contains substantial biomedical domain knowledge, which might not be well captured by general PLMs (Gu et al., 2021). As a result, it remains unknown whether general Text2SQL model is adequate to address Criteria2SQL task.

To study this problem, we create Crit2SQL, a novel and large-scale Criteria2SQL dataset. Crit2SQL contains 4000 pairs of manually-labelled criteria and SQL, making it the largest criteria2SQL datasets annotated by humans compared to two existing datasets Criteria2SQL (Yu et al., 2020) and MIMICSQL (Wang et al., 2020). During the annotation process, we explicitly reveal the generated SQL results by Codex (Chen et al., 2021) to each annotator and ask each annotator to assess the correctness of Codex generation. If the Codex generation is incorrect, the annotator will classify the mistake into one of the seven error types and then annotate the correct SQL query. This annotation process sheds lights on the limitation of Codex, enabling an in-depth analysis of the generation by general PLMs and suggesting directions for future improvement.

Based on this dataset and our analysis, we develop a novel Criteria2SQL approach. The key

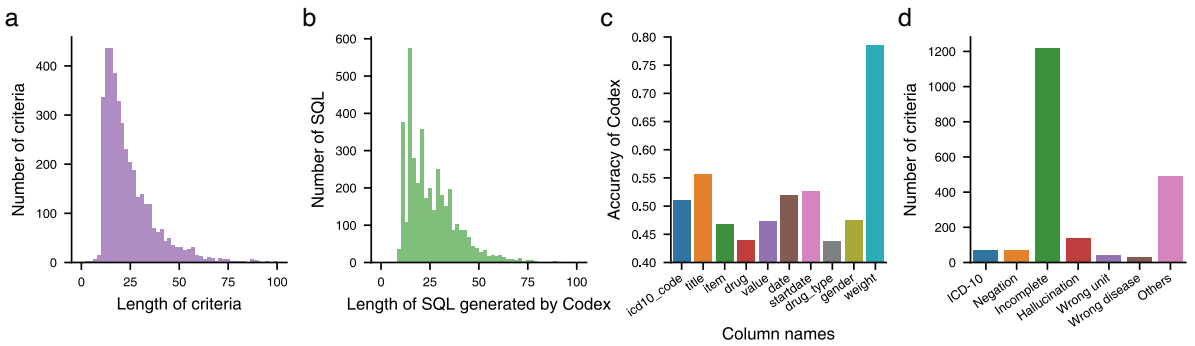

Figure 1: **Analysis of our Crit2SQL Dataset and Codex generation on our dataset. a,** Distribution of the criteria length in our dataset. **b,** Distribution of the length of SQL generated by Codex. **c,** Accuracy of Codex on different column names. **d,** Distribution of seven different error types made by Codex.

idea of our method is to utilize the unique property of Criteria2SQL task, where criteria in the same trial are grouped into inclusion and exclusion criteria. Specifically, we exploit contrastive learning to ensure that inclusion criteria are similar to other inclusion criteria and dissimilar to exclusion criteria in the same trial. We found that our approach substantially outperformed two existing Criteria2SQL methods (Yu et al., 2020; Wang et al., 2020) on our dataset by 11.72% in accuracy. The embedding space of our space further demonstrate that criteria in the same trial are more likely to be clustered together. In summary, we have proposed a novel dataset which contains 4,000 manually annotated SQLs. We have conducted in-depth analysis of common errors made by Codex, suggesting the importance of incorporating domain knowledge into the generation. Finally, our novel contrastive-learning-based approach for it demonstrate the effectiveness of separately modeling inclusion and exclusion criteria.

2 Crit2SQL Dataset

We propose a novel large-scale Criteria2SQL dataset Crit2SQL in this paper. To construct Crit2SQL, we collected 625 real-world clinical trials from ClinicalTrials.gov, containing 4,000 criteria. We excluded criteria that have shorter than 10 tokens without negation and finally obtained 5647 criteria. The criteria are in general long clause, where 15.6% of criteria have more than 35 tokens (**Fig. 1a**). We recruited 13 annotators to manually annotate the SQL query for each criterion. All of them are familiar with SQL and database knowledge.

PLMs have shown impressive results on Text2SQL. To further analyze their performance on our Crit2SQL dataset, we first ran Codex on all of our criteria and generated the corresponding SQL queries. We then demonstrated these commands to annotators. Annotators were asked to first determine whether Codex had generated the correct SQL query. If not, annotators were asked to write down the correct SQL query. Since Codex can often generate reasonable SQL, even if the generation is incorrect, explicitly showing the SQL query to annotators can accelerate the annotation process and enhance the annotation accuracy. After the annotation is finished, we asked each annotator to briefly summarize the common errors made by Codex. We then aggregate these summaries and define 7 common error types made by Codex. Since Codex is not developed specifically for Criteria2SQL generation, these seven error types shed lights on the task-specific challenges and suggest future directions.

3 Common errors by Codex

We first found that Codex obtained a desirable performance on our dataset with a 48.15% accuracy. For those ones that Codex didn't generate correctly, the normalized edit distance between the generated one and the manually-annotated one is also relatively small. For example, more than 37.8% of Codex incorrect generation have less than 0.1 normalized edit distance (**Fig. 2a**).

After observing the overall satisfied results by Codex, we next sought to investigate the incorrect generations by Codex. We first noticed that Codex tends to generate shorter SQL query then the ground-truth one, suggesting the omitting of important information (**Fig. 1b, Fig. 2b**). For ex-

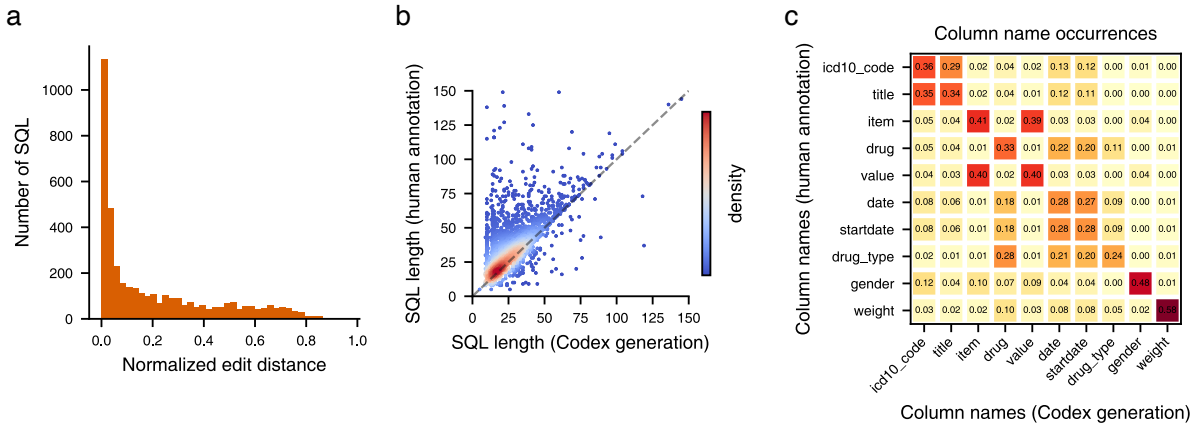

Figure 2: **Analysis of Codex's generation. a,** Normalized edit distance between Codex generation and human annotation. **b,** Comparison between the length of the annotated SQL and the length of the Codex-generated SQL. **c,** Comparison between the column names in SQL generated by Codex and in SQL annotated by human. Each cell in the matrix denotes the percentage of Codex generations that are the same as human annotations.

ample, 55.5% of SQL generated by Codex has less than 25 tokens. In contrast, 68.5% of the human annotation has more than 25 tokens. This substantial discrepancy motivates us to analyze the specific column names that Codex are more likely to omit.

To this end, we calculated the accuracy for frequently appeared column names in our dataset (**Fig. 1c**). The bar of each column name contains all criteria that contains this column name. One criteria might be counted in multiple bars. We found that Codex is most accurate in column names related to *weight*, which is a general column name in many other databases that Codex might have been trained on, indicating the superior performance of Codex in general domains. In contrast, on column names that are specific to Criteria2SQL, the accuracy of Codex is substantially lower. For example, Codex only has 44.0% accuracy on *drug type* and 43.7% accuracy on *drug*. The inadequate performance of Codex on column names related to clinical studies necessities the incorporation of domain-specific knowledge rather than simply adopting a general Text2SQL model.

We next examined the mis-generation between column names to further understand the column names that Codex will incorrectly generate, especially for domain-specific column names (**Fig. 2c**). We found that 11% Codex generations contained *drug type* column names when the ground truth contained *drug* column names and 28% contained *drug* column names when the ground truth had *drug type* column names, suggesting an adequate understanding of domain-specific information. Codex also struggled with *ICD-10* code, which is often incorrectly generated to *date* and *startdate* that also consist of digits. Finally, we summarized Codex's errors into seven error types based on annotators' feedback (**Fig. 1d**). The most common type is the incomplete generation, which accounts for 58.9% of the total errors. We provided a detail analysis of seven error types below.

Wrong ICD-10 code. The first type of error is the generation of wrong ICD-10 code (**Table 1**). The input criteria contains a disease name (e.g., *Persistant atrial fibrillation*), Crteria2SQL model needs to first identify that this is a disease and then identify the corresponding ICD-10 code from it. We found that Codex can often successfully identify that it is a disease, but might map the disease to a wrong ICD-10 code. In this example, Codex mapped *Persistant atrial fibrillation* to the ICD-10 code of another disease *Atrial fibrillation and flutter*, which is the parent node of the correct disease name. One possible solution to address this problem is to use a look-up table to verify the generated ICD-10 code for the disease name. Entity normalization tools are also critical to map a mention to the correct entity in the ICD-10 ontology (Sung et al., 2020).

Negation error. The second type of error is the omitting of negations in the generated SQL (**Table 1**). The input criteria contains negation phrases such as 'don't' and 'not', while Codex is not able to generate the negation SQL keyword 'NOT'. One possible solution to address negation is to apply existing NLP methods for negation detection (Mehrabi et al., 2015) and then incorporate

Error type	Criterion	Codex generation	Human annotation
ICD-10 code	Persistant atrial fibrillation	SELECT subject_id FROM DIAGNOSES_ICD WHERE icd10_code = 'I48'	SELECT subject_id FROM DIAGNOSES_ICD WHERE icd10_code = 'I48.1'
Negative	Don't suffer from granulomatous conditions, liver disease, or kidney disease	SELECT subject_id FROM DIAGNOSES_ICD d INNER JOIN D_ICD_DIAGNOSES i ON d.icd10_code = i.icd10_code WHERE (i.title = 'granulomatous conditions' OR i.title = 'liver disease' OR i.title = 'kidney disease')	SELECT subject_id FROM DIAGNOSES_ICD d INNER JOIN D_ICD_DIAGNOSES i ON d.icd10_code = i.icd10_code WHERE NOT (i.title = 'granulomatous conditions' OR i.title = 'liver disease' OR i.title = 'kidney disease')
Incomplete	History of nephropathy, neuropathy, or retinopathy within 6 months of screening.	SELECT subject_id FROM DIAGNOSES_ICD d INNER JOIN D_ICD_DIAGNOSES i ON d.icd10_code = i.icd10_code WHERE (i.title = 'nephropathy' OR i.title = 'neuropathy' OR i.title = 'retinopathy')	SELECT subject_id FROM DIAGNOSES_ICD d INNER JOIN D_ICD_DIAGNOSES i ON d.icd10_code = i.icd10_code WHERE (i.title = 'nephropathy' OR i.title = 'neuropathy' OR i.title = 'retinopathy') AND NOW()-d.startdate < 180
Hallucination	ALT and/or AST greater than three times the upper limit of the normal range.	SELECT subject_id FROM LABEVENTS WHERE item = 'ALT' AND value > 3*upper_limit_of_normal_range UNION SELECT subject_id FROM LABEVENTS WHERE item = 'AST' AND value > 3*upper_limit_of_normal_range	SELECT subject_id FROM LABEVENTS WHERE item = 'ALT' AND value > 3*uln UNION SELECT subject_id FROM LABEVENTS WHERE item = 'AST' AND value > 3*uln
Wrong unit	Diagnosed with type 2 diabetes mellitus for at least 90 days prior to day of screening	SELECT subject_id FROM DIAGNOSES_ICD WHERE icd10_code = 'E11' AND startdate < NOW()-90*24*60*60	SELECT subject_id FROM DIAGNOSES_ICD WHERE icd10_code = 'E11' AND startdate < NOW()-90
Wrong disease	Blood dyscrasias or any disorders causing hemolysis or unstable red blood cells (malaria, babesiosis, haemolytic anaemia).	SELECT subject_id FROM DIAGNOSES_ICD d INNER JOIN D_ICD_DIAGNOSES i ON d.icd10_code = i.icd10_code WHERE (i.title = 'blood dyscrasias' OR i.title = 'disorders causing hemolysis or unstable red blood cells')	SELECT subject_id FROM DIAGNOSES_ICD d INNER JOIN D_ICD_DIAGNOSES i ON d.icd10_code = i.icd10_code WHERE (i.title = 'blood dyscrasias' OR i.title = 'malaria' OR i.title = 'babesiosis' OR i.title = 'haemolytic anaemia')
Others	Have a blood disorder that would interfere with the drawing of blood glucose measurements or lab samples.	SELECT subject_id FROM LABEVENTS WHERE item = 'Hemoglobin' AND value < 10	SELECT subject_id FROM DIAGNOSES_ICD d INNER JOIN D_ICD_DIAGNOSES i ON d.icd10_code = i.icd10_code WHERE i.title = 'blood disorder'

Table 1: Seven types of errors in Codex generation summarized by our annotators. For each type of error, we show an example criterion, its Codex generation and the human-annotated SQL.

such information into SQL query.

Incomplete error. The third type of error is the incomplete generation where a clause in a criterion is completely missing in the SQL (**Table 1**). This often occurs in long criteria. Since 13% of criteria in our dataset have more than 50 tokens, incomplete error results in a substantial performance decrease by Codex. To address incomplete generation, one can split a long criterion into multiple short ones, separately generate SQL for each one, and then recombine the generated SQL queries.

Hallucination error. The four type of error is the hallucination error, where Codex generates column names that are not mentioned in the original criterion. Hallucination has recently been observed in the text generation of PLMs (Ji et al., 2022). Here, we found that PLM can also induce hallucination in the generation of SQL queries. Among the seven error types, this hallucination error might become even more severe when the language model becomes larger in the future.

Wrong unit. The fifth type of error is the wrong unit generated by Codex, such as wrong date and wrong drug dosage. We found that Codex might incorrectly convert a day unit to seconds, which cannot be aligned with the unit in the table. Understanding and processing numerical values by PLMs is an emerging topic (Geva et al., 2020; Lewkowycz et al., 2022). One possible solution is to detect outlier numbers (e.g., extremely large days) by comparing Codex generation to five-number summary in numerical columns.

Wrong disease. The sixth type of error is the generation of a wrong disease name that cannot be mapped to a column value in the database. This type of error is related to the wrong ICD-10 code error. Different from the wrong ICD-10 code error, such disease names often cannot be exactly matched to any ICD-10 disease, thus necessitating the development of a disease synonym prediction model. Moreover, this type of error is also related to the hallucination error, since Codex might incorrectly combine and regard two short disease names as a very long disease name.

Combination of multiple errors. In addition to the above six major error types, there are some other less frequent error types made by Codex. These errors are often a combination of multiple types of errors. For example, we showed an example where Codex generates the wrong disease name and induces hallucination error (**Table 1**). When a criterion is complicated and might trigger multiple

errors, Codex's generation could be very bad and hard to fix using simple rules or post-processing. This indicates the importance to let clinician write high-quality criteria that follow certain templates and rules.

4 Methods

The encoder-decoder Transformer model T5 has demonstrated great influential in a wide range of tasks (Raffel et al., 2020). By modelling the Text-to-SQL task as machine translation, we were able to directly leverage T5. Contrastive learning have also shown promising results due to its power in reshaping the embedding space by capturing similarities between samples(Chen et al., 2020). Here we developed our Crit2SQL method by incorporating contrastive learning into T5 based on the intuition that inclusion criteria in the same one trial are similar.

4.1 Prompts for Criteria2SQL

Prompts design is vital in providing language models with sufficient guidance to produce relevant and coherence response. To prompt the T5 model for generating correct SQL, we need to specify the potential database tables and the column names as well as criteria used to generate SQL queries. For each table, the prompt template is set as $\mathcal{T}_i$= "CREATE TABLE T_i: $c_1, c_2, \ldots, c_{n_i}$", where T_i represents the table name and $\{c_j\}_{j=1}^{n_i}$ denote the n_i column names of table T_i. Here, the table names in our database are T_I="D_ICD_DIAGNOSES", T_P="PATIENTS", T_L="LABEVENTS", T_D="DIAGNOSES_ICD" and T_{PR}="PRESCRIPTIONS", following real-world electronic medical record datasets MIMIC(Johnson et al., 2016). We combine all the 5 table prompt templates and the criteria R as the prompt for SQL generation as "$\mathcal{T}_I$; $\mathcal{T}_P$; $\mathcal{T}_L$; $\mathcal{T}_D$; $\mathcal{T}_{PR}$. Criteria to SQL: R".

4.2 Fine-tuning T5

Several recent studies (Rajkumar et al., 2022; Lin et al., 2020; Li et al., 2020) formulate Criteria2SQL as a machine translation task, where the source language is the criteria and the target language is the SQL queries. Here we choose the T5 architecture, which obtains state-of-the-art performance on multiple natural language tasks, including machine translation. We fine-tune the encoder-decoder T5 model on our annotated dataset with 2,666 criteria-SQL pairs for training. We use the prompts introduced in Section 3.1 as the input of the encoder during both training and inference. During training, we feed the tokenized SQL as the target to the decoder followed by the auto-regressive loss.

Let the tokenized SQL be a sequence of tokens $y_i = (s_1, s_2, \ldots, s_{l_i})$, where l_i denotes the length of y_i. Then the fine-tuning process learns to predict the next tokens based on previous tokens by minimizing the following log-likelihood (1):

$$\mathcal{L}^{(T5)} = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \sum_{j=1}^{l_i} \mathbb{P}(s_j | s_{j-1}, s_{j-2}, \ldots, s_1), \tag{1}$$

where $\mathcal{S} = \{y_i\}_i$ is the set of all SQL annotations.

4.3 Criteria embedding using contrastive learning

One key observation of the clinical criteria in our dataset is that two criteria within one trial tend to be similar if they are both inclusion criteria or both exclusion criteria. Based on this observation, we exploit contrastive learning to minimize the distance between similar criteria in the embedding space of the encoder of T5. Let $\mathbf{x}^i$ be the input of the i-th criterion with the designed prompt template and Enc be the encoder of T5, the encoder output can be denoted as: $\mathbf{z}^i = Enc(\mathbf{x}^i) = (\mathbf{z}_1^i, \ldots, \mathbf{z}_{l_i}^i)$, where l_i is the length of input $\mathbf{x}^i$ and $\mathbf{z}_j^i$ represents the embedding of the j-th token in $\mathbf{x}^i$. By introducing one extra multi-head attention layer $\mathrm{MultiHeadAttn}()$and one linear layer $\mathrm{Linear}()$, We define the sentence embedding of $\mathbf{x}^i$ as: $\widetilde{\mathbf{z}}^i = \mathrm{Linear}(\mathrm{MultiHeadAttn}(\mathbf{z}^i)[0])$.

Let $\mathcal{N}(i)$ be the index set of samples that are in the same trial and have the same "inclusion" type or "exclusion" type with the i-th sample. Therefore, the i-th sample and j-th sample are defined as a positive pair in our contrastive learning framework if $j \in \mathcal{N}(i)$. Randomly shuffled batches with finite sizes are likely to result in the lack of positive pairs. To mitigate this, we only randomly choose half of the samples in a batch and decide the other half by selecting j from $\mathcal{N}(i)$. To avoid extreme cases where $\mathcal{N}(i) = \varnothing$ and ensure the construction of positive pairs for each i in the first half batch, we expanded $j \in \mathcal{N}(i)$ to $j \in \widetilde{\mathcal{N}}(i)$ where $\widetilde{\mathcal{N}}(i) = \mathcal{N}(i) \bigcup i$. Due to the dropout mechanism in T5 model, the case of $i = j$ is equivalent to the contrastive learning strategy in SimCSE (Gao et al., 2021). Our contrastive learning loss could be

formulated as:

$$\mathcal{L}^{(CL)} = \sum_{i=1}^{n_{batch}} \log(\frac{\sum_{j=1}^{n_{batch}} I_j \exp(\widetilde{\mathbf{z}}^i \cdot \widetilde{\mathbf{z}}^j)}{\sum_{k=1}^{n_{batch}} \exp(\widetilde{\mathbf{z}}^i \cdot \widetilde{\mathbf{z}}^k)}), \quad (2)$$

where $I_j = 1$ when $j \in \widetilde{\mathcal{N}}(i)$. We finally combine the contrastive learning loss with the auto-regressive loss:

$$\mathcal{L}^{(Crit2SQL)} = \mathcal{L}^{(T5)} + \lambda \mathcal{L}^{(CL)}, \quad (3)$$

where λ is a hyperparameter.

5 Results

5.1 Experimental setting

We used 3-fold cross-validation to evaluate our methods. We randomly split the dataset into 3 folds, where 2,666 pairs among the entire 4,000 pairs were used as training and 1,334 pairs were used as test. We evaluated two experimental settings. In addition to the vanilla cross-validation, we also evaluated on criteria SQL pairs where Codex were not able to correctly annotate. For each fold, we chose the criteria that Codex generated incorrectly as the test set, resulting in the test set of 688, 675, and 674 respectively. This task evaluates the capability of correcting and complementing Codex.

Comparison approaches. We compared our method with three text-to-sql approaches. **T5** model has been used as the SQL generation frameworks in previous researches. Here we fine-tuned T5 on our curated datasets with the same training parameters as our model. Since "T5-base" model requires large GPU memory that might not be accessible in real-world clinics, we used the "T5-small" architecture in our study with the hidden embedding dimension of 512. To ensure a fair comparison, we also used this "T5-small" model to initialize our approach. **Criteria2SQL** is a grammar based model that transforms the text-to-sql task into the slots filling task (Yu et al., 2020). It generated slots according to the grammar, and used LSTM (Hochreiter and Schmidhuber, 1997) decoders to predict words for the slots. **TREQS** is a domain-specific method developed to convert questions from electronic medical records (EMR) (Wang et al., 2020). It formulated this task as Seq2Seq machine translation. TREQS adopted the bidirectional LSTM layer as the encoder and one undirectional LSTM layer as the decoder. It also utilized the temporal attention and dynamic attention to further improve the performance. When evaluating the performances, we considered the predictions were correct only when they exactly match the ground truth. Therefore, the calculated prediction accuracy is a lower bound of the true errors.

Training details. We set the hyperparameter λ to 0.5 in (3) based on cross-validation. We performed grid search for learning rate in $[1e^{-4}, 1e^{-2}]$, the batch size in $[4, 16]$ and the epochs in $[10, 30]$. We finally trained our method and T5 baseline with a learning rate of $1e^{-3}$, a batch size of 16 and the epochs of 20. The T5 model was initialized by one released checkpoints "t5-small" in Huggingface (Wolf et al., 2019).

5.2 Improved performance when evaluating on all criteria

We summarized the prediction accuracy of our method and baselines in **Fig. 3a**. Our method outperformed all three comparison approaches, indicating the effective integration of T5 and the trial-based contrastive learning strategy. Our method, T5 and TREQS all formulate Criteria2SQL as machine translation. These three methods achieve significant improvements compared to Criteria2SQL, indicating that Criteria2SQL should be formulated as a Seq2Seq task rather than grammar-based slot prediction.

We next stratified the results based on the tables queried by SQL. We observed that our method achieved a higher performance across all tables than other approaches, indicating the consistent improvements of our method. For example, T5 model does not have a good performance on table D_ICD_DIAGNOSES and PRESCRIPTIONS, while our model substantially improved on these two tables. Both our method and T5 had the highest accuracy on the PATIENTS table, that is significantly higher than other tables. To understand this, we examined the length of SQL that query PATIENTS table and found that the length of SQL containing PATIENTS was significantly shorter than of other SQL queries (p-value$< 3.78e^{-9}$).

5.3 Improved performance on correcting Codex mistakes

We next sought to evaluate our method in the more challenging setting of correctly generating SQL on criteria where Codex made mistakes (**Fig. 3c**). Our methods still outperformed Criteria2SQL and TREQS substantially and slightly improved com-

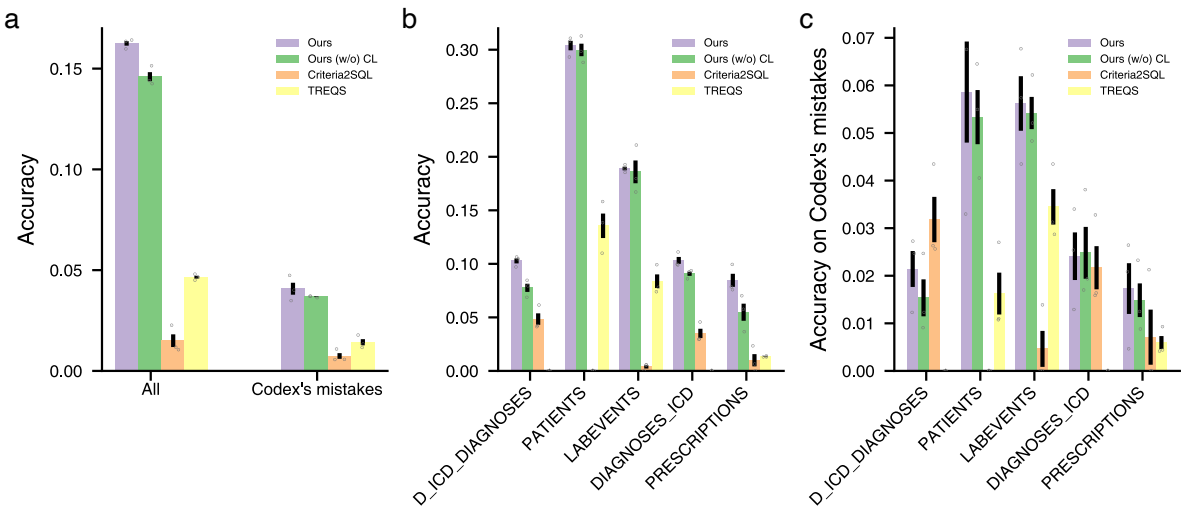

Figure 3: **Performance on Criteria2SQL generation.** **a,** Comparison between our method and comparison approaches (Criteria2SQL, TREQS) on all criteria in our dataset (All) and criteria that Codex generated incorrectly (Codex's mistakes). Ours (w/o) CL denote our methods without the contrastive learning component, which degenerates to T5. **b,c,** Break down performance on different table names evaluating on all criteria in our dataset (**b**) and criteria that Codex generated incorrectly (**c**). Error bar represents the standard error over 3-fold cross validation and its center is the mean value.

pared to T5. This again confirmed the power of PLMs, indicating the future direction of combining contrastive learning with PLMs. The performance of all methods dropped substantially compared to the previous setting, indicating the difficulties of generating SQL queries for these criteria.

5.4 Criteria embedding space for SQL generation

To study how the contrastive learning strategy optimizes the embedding space of the transformer encoder, we visualized both the encoder embedding space of Crit2SQL and T5 in **Fig. 4**. We found that criteria from the same trial were likely to be clustered in the same cluster. In contrast, in the embedding space of T5, these criteria were more evenly distributed **Fig. 4b**. In **Fig. 4c**, we colored samples according to whether the generated SQL queries are correct. We found the criteria demonstrated visible patterns based on the model prediction. Taken together, the criteria embedding space further reveals the improvements of our approach.

6 Related Work

The emerging Criteria-to-SQL task is both challenging and crucial. Promising techniques related to Criteria-to-SQL has the potential to overcome the time-consuming process of recruiting patients in a clinical trial (Wang et al., 2020; Yu et al., 2020). Existing approaches can be divided into two categories as semantic parsing and Seq2Seq models.

Semantic parsing. Semantic-parsing-based methods first convert the input question to a structured tree (Yu et al., 2018b). Previous works have explored the application of semantic parsing (Berant et al., 2013; Poon and Domingos, 2009; Kamath and Das, 2018) to map the unstructured natural language to its logic forms (Hwang et al., 2019; Yu et al., 2018a; Xu et al., 2017). IRNet first bridged the question and database together, and then generate the intermediate representation, based on which the SQL was generated (Guo et al., 2019). There are also methods manually designing templates based on grammar rules. Criteria2SQL defined 11 grammar rules and designed 6 slots for condition number, column name and combination operators (Yu et al., 2020).

Seq2Seq models. Many text-to-SQL approaches have been developed based on Seq2Seq (Sutskever et al., 2014) models by implicitly learning the grammars of SQL queries (Sun et al., 2018; Krishnamurthy et al., 2017). Unlike semantic parsing methods, Seq2Seq models directly generate the target without requiring the predefined slot templates (Wang et al., 2020). The recent development of language generation model, such as BERT (Devlin et al., 2018), GPT2 (Radford et al., 2019), GPT3 (Brown et al., 2020), T5 can also be used for SQL generation. Among them, Codex (Chen et al., 2021), achieved a high accuracy without fine-

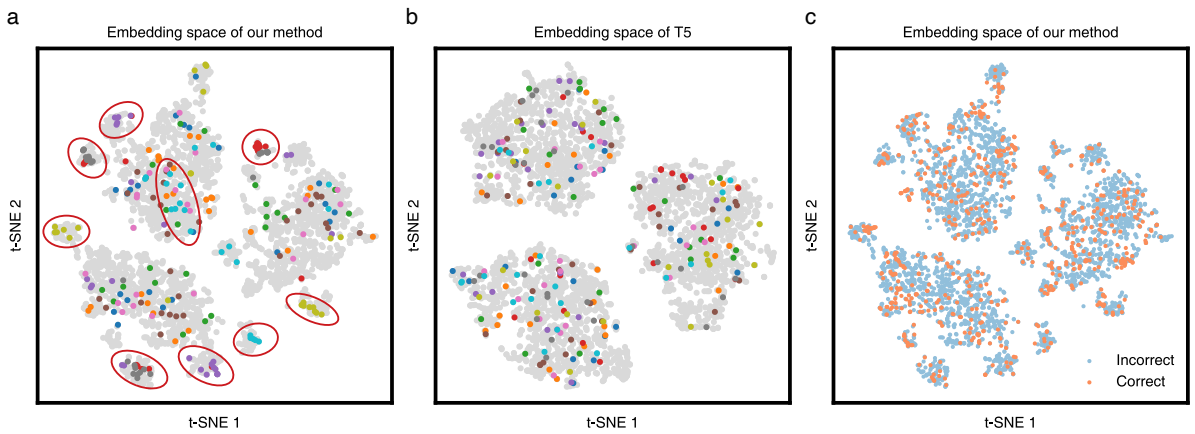

Figure 4: **Visualization of the embedding space by our method and T5. a,b,** Embedding space of our method (**a**) and T5 (**b**). Each point is a criterion. Criteria from the same trial are denoted by the same color. Red ovals demonstrate criteria that are clustered together by our method. **c**, Embedding space of our method colored by correct and incorrect generation.

tuning on downstream text-to-sql dataset.

Text2SQL datasets. Several Text2SQL datasets with have been curated for a wide range of applications, such as ATIS for flight booking (Price, 1990), GeoQuery (Zelle and Mooney, 1996) for geography related questions, WikiSQL for quesitons related to Wikipedia (Zhong et al., 2017). Recently, the biomedical domain specific Criteria2SQL datasets have been proposed to retrieve patients for one clinical trial. For example, MIMICSQL (Wang et al., 2020) collected Question-SQL pairs from the Medical Information Mart for Intensive Care III dataset (Johnson et al., 2016). TREQS offered a large Criteria2SQL dataset for 10,000 criteria (Wang et al., 2020). However, their SQL queries are simulated rather than manually annotated, thus can be limited for real-world application. Yu et al. also created a manually-annotated dataset, which contains 1,983 pairs of criteria and SQL queries. Our dataset is larger than Criteria2SQL, presenting the exsting largest manually-labelled Criteria2SQL dataset.

7 Conclusion and Future Work

We have presented a novel dataset Crit2SQL for studying the task of generating SQL queries for clincal trial criteria. We have recruited 13 annotator to annotate 5647 criteria obtained from ClinicalTrials.gov, offering the existing largest manually-annotated Criteria2SQL dataset. We applied Codex to our dataset and provided an in-depth analysis of codex's generation. We categorized codex's errors into seven types, pointing out future directions to improve Criteria2SQL generation. Based on our observation, we have developed a novel contrastive-learning-based Criteria2SQL approach and obtained substantial improvement against two existing Criteria2SQL approaches.

Our dataset could trigger important future research in Criteria-to-SQL generation. First, it is worth further investigating errors made by PLMs and developed post processing approaches to address them. We have observed that several error types are domain-specific and might be fixed by incorporating biomedical knowledge. Therefore, it might be possible to integrate Codex with biomedical language models, such as PubMedBERT and BioBERT, to consider such domain knowledge.

Second, we noticed that Codex's generation is more accurate when the criteria are well-written. Criteria are often written by clinical who are not familiar with SQL. One future direction is to develop a structured template that lies between SQL and natural language. Instead of developing such templates manually, it might be possible to use PLMs to generate such templates through probing (Alivanistos et al., 2022). We can also incorporate human feedback into the design of such templates.

Finally, an interesting but under-explored task is criteria recommendation. Clinical trial criteria are used to recruit patients from a patient cohort. Too many restricted criteria will substantially reduce the number of selected patients, leading to a failed trial. It might be worth to recommend new loosen criteria to clinician in order to enlarge the current selection and support the validation of the trial. This novel task of criteria recommendation can be studied by integrating our dataset with public electronic medical records, such as MIMIC.

8 Limitations

Our Crit2SQL dataset could be crucial for accelerating the SQL query generation for future clinical studies. However, there are a few limitations we need to address in the future. First, our dataset might not match the column names of another database. It is critical to develop synonym identification approach to map column names in our dataset to other databases. Second, some incorrect generations might be dangerous and select incorrect patients might lead to severe outcomes such as side effects. Therefore, we still need expert validation before applying the generated SQL to patient recruitment.

References

Dimitrios Alivanistos, Selene Báez Santamaría, Michael Cochez, Jan-Christoph Kalo, Emile van Krieken, and Thiviyan Thanapalasingam. 2022. Prompting as probing: Using language models for knowledge base construction. *arXiv preprint arXiv:2208.11057*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Robert M Califf, Deborah A Zarin, Judith M Kramer, Rachel E Sherman, Laura H Aberle, and Asba Tasneem. 2012. Characteristics of clinical trials registered in clinicaltrials. gov, 2007-2010. *Jama*, 307(17):1838–1847.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2021. Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *Computational Linguistics*, 47(2):309–332.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Longxu Dou, Yan Gao, Mingyang Pan, Dingzirui Wang, Jian-Guang Lou, Wanxiang Che, and Dechen Zhan. 2022. Unisar: A unified structure-aware autoregressive language model for text-to-sql. *arXiv preprint arXiv:2203.07781*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. *arXiv preprint arXiv:2004.04487*.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–23.

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. *arXiv preprint arXiv:1905.08205*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2022. Survey of hallucination in natural language generation. *ACM Computing Surveys*.

AE Johnson, Tom J Pollard, Lu Shen, L-w H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, L Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database sci. *Data*, 3(160035):10–1038.

Aishwarya Kamath and Rajarshi Das. 2018. A survey on semantic parsing. *arXiv preprint arXiv:1812.00978*.

George Katsogiannis-Meimarakis and Georgia Koutrika. 2021. A deep dive into deep learning approaches for text-to-sql systems. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2846–2851.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings*

of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1516–1526.

Gyubok Lee, Hyeonji Hwang, Seongsu Bae, Yeonsu Kwon, Woncheol Shin, Seongjun Yang, Minjoon Seo, Jong-Yeup Kim, and Edward Choi. Ehrsql: A practical text-to-sql benchmark for electronic health records. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*.

Ning Li, Bethany Keller, Mark Butler, and Daniel Cer. 2020. Seqgensql–a robust sequence generation model for structured query language. *arXiv preprint arXiv:2011.03836*.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. *arXiv preprint arXiv:2012.12627*.

Hao Liu, Yuan Chi, Alex Butler, Yingcheng Sun, and Chunhua Weng. 2021. A knowledge base of clinical trial eligibility criteria. *Journal of Biomedical Informatics*, 117:103771.

Laura C Lovato, Kristin Hill, Stephanie Hertert, Donald B Hunninghake, and Jeffrey L Probstfield. 1997. Recruitment for controlled clinical trials: literature summary and annotated bibliography. *Controlled clinical trials*, 18(4):328–352.

Saeed Mehrabi, Anand Krishnan, Sunghwan Sohn, Alexandra M Roch, Heidi Schmidt, Joe Kesterson, Chris Beesley, Paul Dexter, C Max Schmidt, Hongfang Liu, et al. 2015. Deepen: A negation detection system for clinical text incorporating dependency relation into negex. *Journal of biomedical informatics*, 54:213–219.

Youcheng Pan, Chenghao Wang, Baotian Hu, Yang Xiang, Xiaolong Wang, Qingcai Chen, Junjie Chen, Jingcheng Du, et al. 2021. A bert-based generation model to transform medical texts to sql queries for electronic medical records: Model development and validation. *JMIR Medical Informatics*, 9(12):e32698.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 1–10.

Patti 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*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*.

Yibo Sun, Duyu Tang, Nan Duan, Jianshu Ji, Guihong Cao, Xiaocheng Feng, Bing Qin, Ting Liu, and Ming Zhou. 2018. Semantic parsing with syntax- and table-aware sql generation. *arXiv preprint arXiv:1804.08338*.

Mujeen Sung, Hwisang Jeon, Jinhyuk Lee, and Jaewoo Kang. 2020. Biomedical entity representations with synonym marginalization. *arXiv preprint arXiv:2005.00239*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Tony Tse, Rebecca J Williams, and Deborah A Zarin. 2009. Reporting "basic results" in clinicaltrials. gov. *Chest*, 136(1):295–303.

Lihan Wang, Bowen Qin, Binyuan Hui, Bowen Li, Min Yang, Bailin Wang, Binhua Li, Jian Sun, Fei Huang, Luo Si, et al. 2022. Proton: Probing schema linking information from pre-trained language models for text-to-sql parsing. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1889–1898.

Ping Wang, Tian Shi, and Chandan K Reddy. 2020. Text-to-sql generation for question answering on electronic medical records. In *Proceedings of The Web Conference 2020*, pages 350–361.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

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

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. *arXiv preprint arXiv:1804.09769*.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. Syntaxsqlnet: Syntax tree networks for complex and cross-domaintext-to-sql task. *arXiv preprint arXiv:1810.05237*.

Xiaojing Yu, Tianlong Chen, Zhengjie Yu, Huiyu Li, Yang Yang, Xiaoqian Jiang, and Anxiao Jiang. 2020. Dataset and enhanced model for eligibility criteria-to-sql semantic parsing. In *12th International Conference on Language Resources and Evaluation (LREC)*.

Deborah A Zarin, Tony Tse, Rebecca J Williams, Robert M Califf, and Nicholas C Ide. 2011. The clinicaltrials. gov results database—update and key issues. *New England Journal of Medicine*, 364(9):852–860.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

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