# *Know What I don't Know*: Handling Ambiguous and Unanswerable Questions for Text-to-SQL

**Bing Wang**[†][*], **Yan Gao**[§], **Zhoujun Li**[†], **Jian-Guang Lou**[§]
[†]Beihang University, [§]Microsoft Research Asia
`{bingwang, lizj}@buaa.edu.cn, {yan.gao, jlou}@microsoft.com`

## Abstract

The task of text-to-SQL is to convert a natural language question to its corresponding SQL query in the context of relational tables. Existing text-to-SQL parsers generate a "plausible" SQL query for an arbitrary user question, thereby failing to correctly handle problematic user questions. To formalize this problem, we conduct a preliminary study on the observed ambiguous and unanswerable cases in text-to-SQL and summarize them into 6 feature categories. Correspondingly, we identify the causes behind each category and propose requirements for handling ambiguous and unanswerable questions. Following this study, we propose a simple yet effective counterfactual example generation approach for the automatic generation of ambiguous and unanswerable text-to-SQL examples. Furthermore, we propose a weakly supervised model DTE (**D**etecting-**T**hen-**E**xplaining) for error detection, localization, and explanation. Experimental results show that our model achieves the best result on both real-world examples and generated examples compared with various baselines. We will release data and code for future research[1].

| Movie | IMDB Rating | Rotten Tomatoes Rating | Content Rating |
|---|---|---|---|
| Titanic | 7.9 | 86% | 7.6 |
| Avatar | 7.8 | 87% | 7.7 |

(a) **Ambiguous Question:** Show me the top rating movie.
**Previous:** SELECT [Movie] ORDER BY [IMDB Rating] DESC LIMIT 1
**Ours:** Oops, this question has multiple answers. "rating" may refer to either "IMDB Rating", "Rotten Tomatoes Rating", or "Content Rating".

| Brand | Sales | Year |
|---|---|---|
| Toyota | 1,933,099 | 2021 |
| Ford | 1,804,824 | 2021 |

(b) **Unanswerable Question:** Show me model name sorted by sales.
**Previous:** SELECT [Brand] ORDER BY [Sales] DESC
**Ours:** Sorry, we can't find an answer for you since "model name" cannot be mapped to any concepts in your table.

Figure 1: Ambiguous and unanswerable examples in text-to-SQL task as well as our explanations. Blue font denotes the problematic question span and red font means the "plausible" column name selected by previous models.

## 1 Introduction

The goal of Text-to-SQL is to generate an executable SQL query given a natural language (NL) question and corresponding tables as inputs. It builds a natural language interface to the database to help users access information in the database (Popescu et al., 2003), thereby receiving considerable interest from both industry and academia (Guo et al., 2019; Wang et al., 2020; Liu et al., 2021). Correspondingly, a series of new model architectures have been proposed, such as IRNet (Guo et al., 2019), RAT-SQL (Wang et al., 2020), ETA (Liu et al., 2021), etc. These models have achieved satisfying results on popular leaderboards, such as Spider (Yu et al., 2018) and WikiSQL (Zhong et al., 2017).

However, state-of-the-art models trained on the leaderboard datasets still have poor performance in realistic scenarios, where user questions are expressed diversely and sometimes are problematic. Concretely, from our study with real-world text-to-SQL examples (Sec. 2), about 20% of user questions are problematic, including ambiguous and unanswerable questions: (1) *ambiguous*: one question actually can have multiple semantic meanings based on one table. For example in Figure 1 (a), the word *rating* in the user question has multiple mapping relationships with columns "IMDB Rating", "Rotten Tomatoes Rating", and "Content Rating"; (2) *unanswerable*: users might ask questions that can not be answered based on their tables. For example, in Figure 1(b), there is no column about "*model name*" in the table. State-of-the-art models generate a "plausible" SQL query when faced with

[*]Work done during an internship at Microsoft Research Asia.
[1]https://github.com/wbbeyourself/DTE

ambiguous and unanswerable questions.

This phenomenon reveals two problems of previous methods. First, in the data aspect, the training data of their methods lack ambiguous and unanswerable samples. Questions in current datasets for training are collected either by templates (Zhong et al., 2017) or by annotating controlled questions and excluding poorly phrased and ambiguous ones (Yu et al., 2018). This data collection process means a correct answer is guaranteed to exist in the table context. Second, in the model aspect, end-to-end parsing models ignore modeling the questions in a fine-grained manner, which makes it impossible to accurately detect and locate the specific reason for ambiguous or unanswerable questions.

To address the data shortage problem, we propose a counterfactual examples generation approach for automatically generating ambiguous and unanswerable text-to-SQL examples based on existing datasets. As we know, conditional NL modification technologies are not natural and accurate due to the free-form characteristic of text. Compared with plain text, tables have well-defined structures, typically in rows and columns. Therefore table modification is more controllable. In light of this, we propose to generate ambiguous and unanswerable examples by modifying the structured table.

Furthermore, we propose a weakly supervised model DTE (**D**etecting-**T**hen-**E**xplaining) for handling ambiguous and unanswerable questions. To locate ambiguous or unanswerable tokens in user questions, we formulate the location process as a sequence labeling problem, where each token in the user question will be tagged as being related to an ambiguous label, an unanswerable label, or others (Sec. 4.1). Since there is no labeled data for sequence labeling, we extract the set of column names and cells appearing in the SQL query and use this set as the weak supervision. In this way, we could generate explicit explanations for ambiguous and unanswerable questions to end users. Note that the sequence labeling information is pseudo and derived from our model, thus alleviating heavy manual efforts for annotation.

Experimental results show that our approach achieves the best results on both real-world examples collected from realistic applications and automatically generated ambiguous and unanswerable examples, compared with various baselines. Our contributions are as follows:

- We conduct a preliminary study on the ambiguous and unanswerable questions in text-to-SQL and summarize 6 featured categories. We also identify the causes behind each category and propose requirements that should be met in explainable text-to-SQL systems.
- We propose a counterfactual examples generation approach for automatically generating ambiguous and unanswerable text-to-SQL examples via modifying both user questions and structured tables.
- We propose a weakly supervised model for ambiguous and unanswerable question detection and explanation. Experimental results show that our approach brings the model with the best explainability gain compared with various baselines.

## 2 Preliminary Study on Ambiguous and Unanswerable Problem

To understand user behaviors in a real-world application, we conduct a comprehensive user study on our commercial text-to-SQL product. Firstly, around 3,000 failed user questions in the product are collected. They obtained over 30 data tables from multiple domains, including education, finance, government, etc. Then, we manually group these questions into multiple categories. At last, we explore the causes and potential solutions to deal with them. According to our analysis, nearly 20% of the questions are problematic, including 55% ambiguous and 45% unanswerable questions respectively, revealing the importance of handling problematic questions. In the following, we will introduce their categories, causes, and potential solutions for handling them.

### 2.1 Problem Categories

In this section, we formalize ambiguous and unanswerable questions and identify 6 sub-categories.

**Ambiguous Problem** In the text-to-SQL task, ambiguity means that one user question could have multiple semantic meanings (e.g., SQL query) based on one table. Specifically, we identify two specific sub-categories, namely column ambiguity and value ambiguity, which account for 45% and 10% of all problematic questions, respectively. Column ambiguity means that some tokens in the user question could be mapped to multiple columns. For example in Table 1, we don't know exactly which "Rating" the user wants since there are three *rating*

| Ambiguous Problem | | |
|---|---|---|
| **Category** | **Example** | **Percentage** |
| Column Ambiguity | Question: Show me the top rating movie.<br>Columns: Movie, IMDB Rating, Rotten Tomatoes Rating, Content Rating<br>Note: The token "rating" in question is ambiguous because there are 3 column names containing "rating". | 45% |
| Value Ambiguity | Question: For Jack, show me the date of license issued and license expires.<br>Columns: Engineer, Constructor, License issued, License expires, …<br>Values: Jack::Engineer, Jack::Constructor<br>Note: The token "Jack" in question is ambiguous because both column "Engineer" and "Constructor" have the same cell value "Jack". | 10% |
| **Unanswerable Problem** | | |
| **Category** | **Example** | **Percentage** |
| Column Unanswerable | Question: Show me model name by sales.<br>Columns: Brand, Sales, Year<br>Note: The span "model name" is unanswerable because no such a column named "model name". | 30% |
| Value Unanswerable | Question: Count the total of Private hospitals.<br>Columns: NHHospitalCategory, State, Year, BenefitsPaid<br>Note: The span "Private hospitals" is unanswerable because no such value in column "NHHospitalCategory". | 7% |
| Calculation Unanswerable | Question: What is the balance of trade of China ?<br>Columns: Country, Imports, Exports, …<br>Note: The span "balance of trade" is unanswerable, because model does not know the calculation formula : Balance of Trade = Exports – Imports. | 6% |
| Out of Scope | Question: Bar chart showing the Word count of every character.<br>Columns: ChapterNo, WordCount, ChScID, SceneName<br>Note: The span "Bar chart" is out of the model's scope, the model does not support the graphic operation. | 2% |

Table 1: Ambiguous and unanswerable problem categories in text-to-SQL task. The red font with a dashed line denotes the ambiguous or unanswerable question span. The green font means a related concept (columns or values) to the red span. Note sentence explains why the example is ambiguous or unanswerable.

columns. Value ambiguity means that some tokens in the user question could be mapped to multiple cell values in the table. For example in Table 1, *Jack* in the user question can be mapped to the name of either an "Engineer" or a "Constructor".

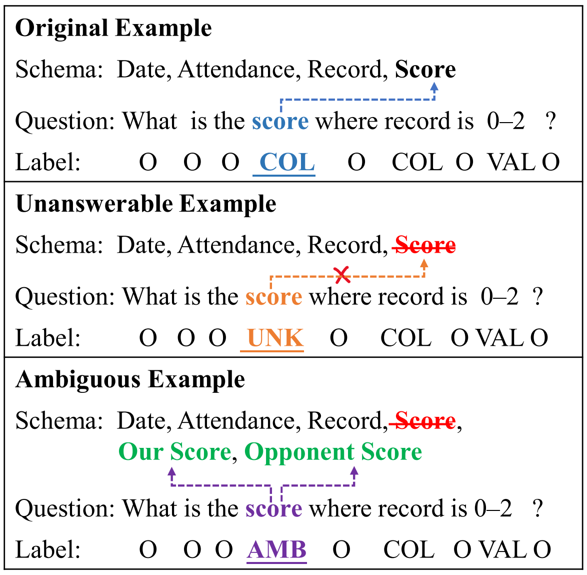
**Unanswerable Problem** The unanswerable problem can be classified into four categories: column unanswerable, value unanswerable, calculation unanswerable, and out-of-scope, which account for 30%, 7%, 6%, and 2% of all problematic questions, respectively, as shown in the bottom part of Table 1. (1) The column unanswerable means that the concepts mentioned in the question do not exist in table columns. In the first example, the *model name* is not existed in the given columns, but our product incorrectly associates it with the irrelevant column "Brand". (2) The value unanswerable indicates that the user question refers to cell values that do not exist in the table. As the second example shows, no such *Private hospitals* value exists in the table. (3) The calculation unanswerable category is more subtle. It requires mapping the concept mentioned in the user question to composite operations over existing table columns. For example, the *balance of trade* is a concept derived from "$Exports - Imports$". Such mapping functions require external domain knowledge. Our product which is trained from a general corpus captures limited domain knowledge, and thus often fails. (4) The out-of-scope category means that the question is out of SQL's operation scope, such as chart operations.

## 2.2 Causes

Through communicating with end users and analyzing the characteristic of questions as well as corresponding table contexts, we identify three fundamental causes for ambiguous and unanswerable questions: (1) end users are unfamiliar with the content of the table and don't read the table carefully, causing unanswerable questions; (2) ambiguity arises due to the richness of natural language expressions and the habitual omission of expressions by users (Radhakrishnan et al., 2020); (3) the emergence of similar concepts in the table tends to cause more ambiguous questions. Note that around 95% of problematic questions are constructed unintentionally, revealing the importance of making users *conscious of being wrong*.

## 2.3 Explainable Parser Requirements

Based on the findings and analysis above, to deal with ambiguous and unanswerable questions, we


**Original Example**

Schema: Date, Attendance, Record, **Score**

Question: What is the **score** where record is 0–2 ?

Label: O O O COL O COL O VAL O

**Unanswerable Example**

Schema: Date, Attendance, Record, ~~**Score**~~

Question: What is the score where record is 0–2 ?

Label: O O O UNK O COL O VAL O

**Ambiguous Example**

Schema: Date, Attendance, Record, ~~**Score**~~, **Our Score**, **Opponent Score**

Question: What is the score where record is 0–2 ?

Label: O O O AMB O COL O VAL O

Figure 2: Ambiguous and unanswerable examples generated by our approach.

propose to make a text-to-SQL system *know-what-I-don't-know*. On one hand, a parsing system should detect ambiguous and unanswerable questions. On the other hand, a parsing system should locate the specific reasons and generate corresponding explanations to guide the user in rectification.

Achieving *know-what-I-don't-know* can benefit from two aspects: (1) from model view: enhances models' ability to deal with problematic questions and improve user trustiness; (2) from user view: makes it clear to users which part of their questions are problematic, guiding them to revise their questions. In our user study experiments, we find that 90% of the problematic questions can be corrected by prompting users with explanations shown in Table 1, and the remaining 10% of questions can only be solved by injecting external knowledge into the model. In the following, we will introduce how we mitigate the challenges mentioned in Sec. 1

## 3 Counterfactual Examples Generation

To alleviate the data shortage issue, we propose a counterfactual examples generation approach for automatically generating problematic text-to-SQL examples. In our approach, we mainly focus on generating two major types of problematic examples: column ambiguity and column unanswerable, which account for 75% of all problematic examples based on our preliminary study. Note that the counterfactual examples are generated via modifying structured tables instead of natural language questions. The reason is that conditional modification on a structured table is more controllable than unstructured text. Finally, 23k problematic examples are obtained based on two text-to-SQL datasets, i.e., WikiSQL (Zhong et al., 2017) and WTQ (Shi et al., 2020). Next, we will introduce the details of our approach.

|  | NoisySP | WikiSQL | WTQ |
|---|---|---|---|
| **Train** |  |  |  |
| # ambiguous | 4,760 | 0 | 0 |
| # unanswerable | 10,673 | 0 | 0 |
| # answerable | 0 | 56,350 | 7,696 |
| # tables | 4,861 | 17,984 | 1,283 |
| **Development** |  |  |  |
| # ambiguous | 1,581 | 0 | 0 |
| # unanswerable | 1,652 | 0 | 0 |
| # answerable | 0 | 8,142 | 1,772 |
| # tables | 1,232 | 2,614 | 325 |
| **Test** |  |  |  |
| # ambiguous | 2,332 | 0 | 0 |
| # unanswerable | 2,560 | 0 | 0 |
| # answerable | 0 | 15,362 | 0 |
| # tables | 1,993 | 5,031 | 0 |

Table 2: Dataset statistics of NOISYSP, compared to the original WikiSQL and WTQ dataset.

### 3.1 Our Approach

Given an answerable text-to-SQL example that contains a question $Q = (q_1, \dots, q_m)$, a DB schema (also a column set) $\mathcal{C} = \{c_1, \dots, c_n\}$ and a SQL query $S$, our goal is to generate problematic examples, denoting as ($Q$, $\mathcal{C}^{'}$, $S$) triplets. By removing evidence supporting $Q$ from $\mathcal{C}$ or adding ambiguous ones, a new DB schema $\mathcal{C}^{'}$ is generated.

**Unanswerable Examples Generation** Specifically, we randomly sample a target column $c_t$ in the SQL query $S$. Then we delete $c_t$ from $\mathcal{C}$ to remove the supporting evidence for question spans that mentioned $c_t$. At last, the above question span is recorded as a UNK label. For instance, in the unanswerable example of Figure 2, given an original question "What is the score where record is 0–2?", the question span "score" is grounded to the column "Score". By deleting the column "Score", we obtain an unanswerable example.

**Ambiguous Examples Generation** Similar to unanswerable examples generation, we generate an ambiguous example by firstly deleting a column $c_t$ and then adding two new columns. The critical point is that newly added columns are expected to (1) fit nicely into the table context; (2) have high semantic associations with the target column $c_t$ yet low semantic equivalency (e.g. "opponent score" is semantic associated with "score", but it is not

| Labels | Description | Example(Token:Label) |
|---|---|---|
| COL | Column Mention | sales: B-COL |
| VAL | Value Mention | godfather: B-VAL |
| AMB | Ambiguous Span | rating: B-AMB |
| UNK | Unanswerable Span | model: B-UNK |
| O | Nothing | the: O |

Table 3: Labeling categories of question tokens

semantic equivalent). To achieve this, we leverage an existing contextualized table augmentation framework, CTA (Pi et al., 2022), tailored for better contextualization of tabular data, to collect the adding column candidates. After that, we rerank the column candidates by their length and similarity with the column $c_t$, and keep the top 2 as our newly added columns. As shown in the ambiguous example of Figure 2, we first delete the original column "Score", then add two domain-relevant and semantically associated columns "Our Score" and "Opponent Score".

## 3.2 Dataset Statistic

Leveraging our counterfactual examples generation approach, we obtain a dataset, called NoisySP based on two cross-domain text-to-SQL datasets, i.e., WikiSQL (Zhong et al., 2017) and WTQ (Shi et al., 2020). Consistent with our preliminary study, we generate 20% of the original data count as problematic examples. Finally, we get 23k problematic examples. Detailed statistics can be seen in Table 2. To ensure the quality of the development set and test set, we hired 3 annotators to check the candidate set of newly added columns for ambiguous examples and then drop low-quality ones. Note that the rate of low quality is only 5%, demonstrating the effectiveness of our approach.

## 4 Model: DTE

In this section, we introduce our **D**etecting-**T**hen-**E**xplaining (DTE) model to handle ambiguous and unanswerable questions. To generate a fine-grained explanation, we formulate it as a sequence labeling problem, where each token in the user question will be tagged as being related to an ambiguous label, an unanswerable label, or others. Concretely, DTE consists of three modules: concept prediction module, grounding module, and sequence labeling module. The grounding module generates pseudo-label information to guide the training of the sequence labeling module. The overall architecture of DTE is shown in Figure 3.

### 4.1 Task definition

Given an input question $Q = (q_1, \ldots, q_m)$, a data table (with a concept set $C = c_1, \ldots, c_k$, containing columns and cell values), the goal of sequence labeling is to output a labeling sequence $L = (l_1, \ldots, l_m)$ for each token in $Q$. It can be represented by tagging each token in the question with a set of BIO labels (Tjong Kim Sang and Veenstra, 1999). Specifically, we define 5 kinds of labels for question tokens, namely COL, VAL, AMB, UNK, and O. Their descriptions and examples are shown in Table 3.

### 4.2 Preliminaries: ETA for grounding

In this work, we formulate problematic question detection as a sequence labeling task, whose training process requires large-scale label annotations as supervision. However, such annotations are expensive and time-consuming. To obtain label information in an efficient and cheap way, we propose to leverage the grounding result of the text-to-SQL task and transform it into a pseudo-labeling sequence. Particularly, we use ETA (Liu et al., 2021), a probing-based grounding model from pretrain language models (PLMs), as the backbone of our approach. The major advantage of ETA is that, compared with models relying on expensive annotations of grounding, it only needs supervision that can be easily derived from SQL queries.

### 4.3 Sequence Labeling Module

To meet the requirements of detecting and locating ambiguous and unanswerable question spans, we design a sequence labeling module, which is intuitively suitable for our sequential modeling purpose. The sequence labeling module consists of a dropout layer, a linear layer, and a CRF layer, following best practices in previous work (Yang et al., 2018). Given a contextualized embedding sequence $(e_{q_1}, \ldots, e_{q_m})$, the goal of the sequence labeling module is to output the label sequence $L = l_1, \ldots, l_m$ with the highest likelihood probability.

### 4.4 Multi-Task Training

Our multi-task training process involves three steps: (1) train the concept prediction module. (2) warm-up grounding module to get alignment pairs. (3) train the sequence labeling module with the pseudo tag derived from the grounding module.

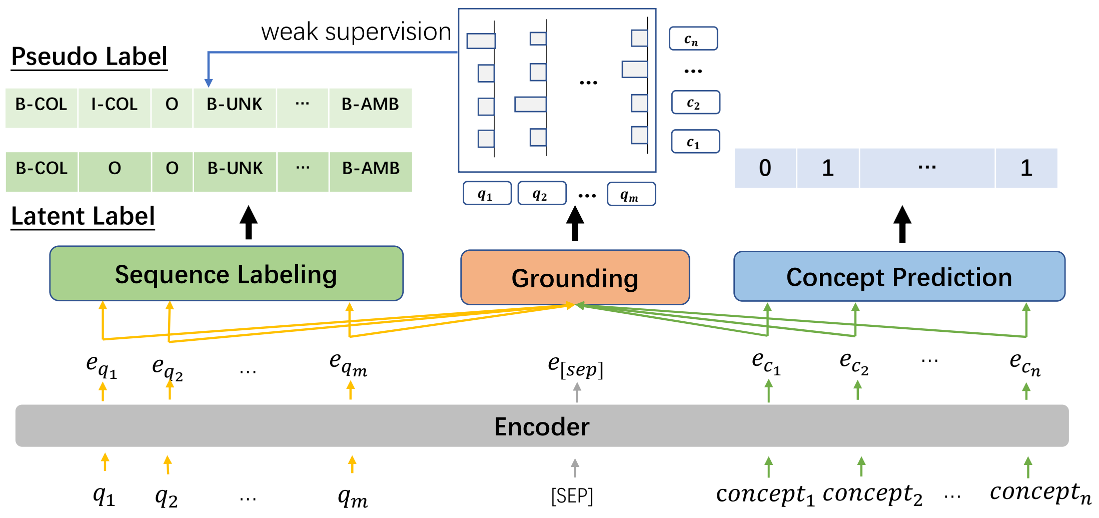


Figure 3: The overall architecture of DTE.

### 4.5 Response Generation

At the inference step, given a question and the table information, DTE predicts labels for each question token and outputs grounding pairs between question tokens and table entities. If the AMB (or UNK) label occurs, it means it is an ambiguous (or unanswerable) question. To generate corresponding interpretations to end users, we carefully design two response templates. More details about the templates could be found in Appendix A.3.

## 5 Experiments

In this section, we systematically evaluate the effectiveness of DTE. Specifically, we examine DTE's performance in two aspects: (1) the performance of the sequence labeling module in detecting ambiguous and unanswerable tokens; (2) the grounding performance for each label to provide evidence for generating explainable responses to end users. In addition, we report the evaluation results on text-to-SQL task.

### 5.1 Experimental Setup

**Datasets** We conduct experiments based on the following datasets: (1) NOISYSP with 23k automatically generated examples, (2) two cross-domain text-to-SQL datasets, i.e., WikiSQL (Zhong et al., 2017) and WTQ (Shi et al., 2020)[2], (3) 3,000 real-world examples collected by ours (Sec. 2). All models are trained with the NOISYSP, WikiSQL and WTQ datasets. Specifically, real-world examples are only used for testing. Dataset statistics are shown in Table 2.

**Evaluation Metric** To evaluate sequence labeling performance, we report accuracy for each label category. For grounding performance evaluation, we report grounding accuracy for each label, except for UNK and O, which have no grounding results.

**Baseline Models** We choose two types of representative models for comparison: (1) the heuristic-based method (Sorokin and Gurevych, 2018), which is widely used in entity linking and grounding tasks; (2) the learning-based method, ETA (Liu et al., 2021), which is a strong grounding baseline, leveraging the intrinsic language understanding ability of pretrained language models. We update them with a little modification to fit our task because their vanilla version is not directly applicable. More implementation details about the baseline and DTE could be found in Appendix A.

### 5.2 Experimental Results on NOISYSP

**Sequence Labeling Results** As shown in Table 4, we compare the performances of DTE with various baselines on the test set of NOISYSP. DTE outperforms previous baselines across all label categories, which demonstrates the superiority of our DTE model. Compared with the heuristic-based method, DTE significantly improves performances by 25% average gains of all label categories, which shows that our NOISYSP dataset is challenging and the heuristic-based method is far from solving these questions. Besides, DTE consistently outperforms the ETA+BERT baseline by a large margin, not only improving the ambiguous and

[2]Note that we use the version with SQL annotations provided by Shi et al. (2020) , since the original WTQ (Pasupat and Liang, 2015) only contains answer annotations.

| Models | COL | VAL | AMB | UNK | O |
|---|---|---|---|---|---|
| Heuristic | 61.7 | 66.8 | 57.8 | 60.7 | 72.1 |
| ETA+BERT | 83.4 | 87.9 | 75.6 | 70.2 | 80.9 |
| ETA+BERT$_L$ | 85.7 | 90.4 | 76.4 | 71.4 | 82.7 |
| DTE +BERT | 88.2 | 94.1 | 81.4 | 78.6 | 90.7 |
| DTE +BERT$_L$ | **89.4** | **95.7** | **83.2** | **80.3** | **92.4** |

Table 4: Sequence labeling accuracy of DTE compared with baselines in NOISYSP test set.

| Models | COL | VAL | AMB |
|---|---|---|---|
| Heuristic | 55.9 | 67.2 | 56.2 |
| ETA+BERT | 71.4 | 75.3 | 60.7 |
| ETA+BERT$_L$ | 72.4 | 77.8 | 62.4 |
| DTE +BERT | 73.4 | 78.2 | 79.8 |
| DTE +BERT$_L$ | **75.1** | **80.7** | **82.4** |

Table 5: Grounding accuracy of DTE compared with baselines in NOISYSP test set.

unanswerable label accuracy by 7% and 11%, respectively, but also improving column and value detecting accuracy, demonstrating the effectiveness of our approach for detection.

**Grounding Results** To locate the specific reasons for ambiguous questions, we are required to not only detect target spans but also find the linked concept (column or value), namely grounding. As shown in Table 5, we compare DTE's grounding performance with various baselines. Note that the unanswerable span in question does not require grounding to any concept, thus we do not report the grounding result of it. We observe that DTE consistently outperforms baselines across three label categories on grounding performance, especially on the ambiguous grounding whose linked concepts are more various and diverse.

## 5.3 Generalization on Realistic Data

To verify the generalization ability of DTE, we conduct *out-of-distribution* experiments on 3k realistic data. As shown in Figure 4, our DTE model still outperforms all baselines consistently and achieves promising performance in ambiguous and unanswerable detection with 75.2% and 70.2% sequence labeling accuracy, respectively. From Figure 5, we observe that our DTE model outperforms other baselines by a large margin in grounding accuracy of ambiguous spans. These results indicate the generalization ability of DTE for handling ambiguous and unanswerable questions and the effectiveness of realistic data.

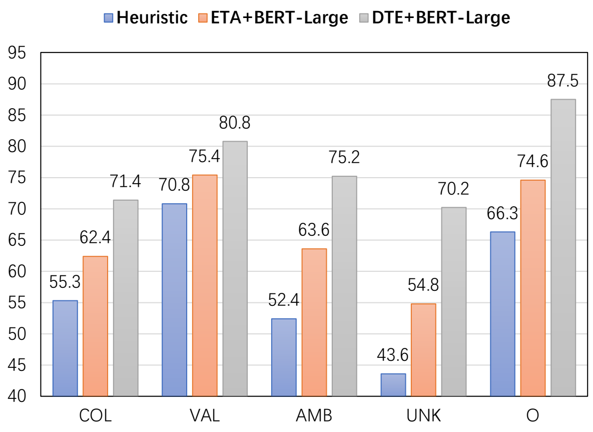


Figure 4: Sequence labeling accuracy of DTE compared with baselines in realistic data.

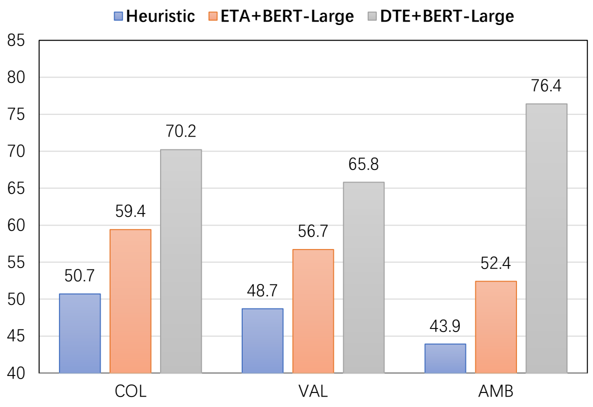


Figure 5: Grounding accuracy of DTE compared with baselines in realistic data.

## 5.4 Text-to-SQL Results

To verify the influence of DTE on the text-to-SQL task, we report the exact match accuracy (Ex.Match) and execution accuracy (Ex.Acc) on WTQ dataset. As shown in Table 6, compared with ALIGN (Lei et al., 2020) and ETA (Liu et al., 2021), DTE shows slightly better performance in both exact match accuracy and execution accuracy. It should be noted that all questions in this experiment are normal questions, i.e., without ambiguous and unanswerable questions. The result shows that DTE can boost text-to-SQL performance instead

| Model | Dev | | Test |
|---|---|---|---|
| | Ex.Match | Ex.Acc | Ex.Acc |
| ALIGN | 37.8 | 56.9 | 46.6 |
| ALIGN+BERT | 44.7 | 63.8 | 51.8 |
| ETA+BERT | 47.6 | 66.6 | 53.8 |
| DTE+BERT | **48.1** | **66.5** | **54.2** |

Table 6: Ex.Match and EX.ACC of text-to-SQL results on the dev and test set of WTQ.

| Models | COL | VAL | O |
|---|---|---|---|
| DTE w/ SL | **75.1** | **80.7** | **90.5** |
| DTE w/o SL | 72.6 | 75.2 | 82.8 |

Table 7: Ablation study of DTE model with BERT-large in the grounding accuracy on the test set of NOISYSP. SL means sequence labeling module.

of damaging it.

### 5.5 Discussion

**What are the remaining errors?** We manually analyze 20% of the remaining errors in the NOISYSP dataset and summarize four main error types: (1) wrong detection (25%) - where our model either misses or over predicts the ambiguous or unanswerable label. (2) widened span (30%) - where our model predicts a longer span than the golden result. (3) narrowed span (25%) - where the model infers a narrowed span than the golden result. (4) other errors (20%) are caused by the grounding module. This error analysis indicates the main challenge of DTE is precise localization rather than detection because the second and third errors (55%) are caused by the wrong span boundary. More detailed examples can be seen in Appendix B.

**Can sequence labeling module benefit grounding module?** As a multi-task training approach, it is critical to determine the effect of introducing extra tasks on the models' performance on original tasks. To verify the influence of the sequence labeling module on grounding results, we conduct an ablation study with or without the sequence labeling module. As we can see in Table 7, the grounding module does achieve better performance on columns and values alignment with the sequence labeling module. Through our analysis, we find the performance gain mainly comes from *long concept mention* (with token length > 4). The reason is that the CRF layer in the sequence labeling module can strengthen the grounding module's ability to capture long-distance dependency. In summary, we can conclude that the sequence labeling task can fit in well with the grounding task.

## 6 Related Work

**Problematic Question Detection.** Existing works on problematic question detection can be classified into two categories: (1) heuristic-based methods leverage elaborate rules to detect and locate problematic questions span (Sorokin and Gurevych, 2018; Li et al., 2020; Wu et al., 2020), suffering from heavy human efforts on feature engineering. Besides, some approaches (Dong et al., 2018; Yao et al., 2019) estimate the confidence of parsing results, relying on existing parsing models; (2) on the contrary, learning-based methods don't rely on heuristic rules and parsing models. For example, Arthur et al. (2015) jointly transforms an ambiguous query into both its meaning representation and a less ambiguous NL paraphrase via a semantic parsing framework that uses synchronous context-free grammars. Zeng et al. (2020) trains a question classifier to detect problematic questions and then employed a span index predictor to locate the position. However, the index predictor could only locate at most one error span for each example, limiting its usage scenario. In this work, we propose a learning-based approach that could handle multiple errors in problematic questions.

**Uncertainty Estimation.** Recent works on uncertainty estimation of neural networks explore diverse solutions, such as deep ensembles in prediction, calibration, and out-of-domain detection (Liu et al., 2020). However, these methods require changes to the network and optimization process, typically ignore prior knowledge about the data (Loquercio et al., 2020), and can only get the uncertainty of predictions without locating the reasons. In this work, we propose the counterfactual examples generation approach to adding prior knowledge to the training data and then propose a weakly supervised model for problematic span detection and give explainable reasons.

## 7 Conclusion

We investigate the ambiguous and unanswerable questions in text-to-SQL and divide them into 6 categories, then we sufficiently study the characteristics and causes of each category. To alleviate the data shortage issue, we propose a simple yet effective counterfactual example generation approach for automatically generating ambiguous and unanswerable text-to-SQL examples. What's more, we propose a weakly supervised model for ambiguous and unanswerable question detection and explanation. Experimental results verify our model's effectiveness in handling ambiguous and unanswerable questions and demonstrate our model's superiority over baselines.

## Ethics Statement

Our counterfactual examples generation approach generates a synthesized dataset based on two mainstream text-to-SQL datasets, WikiSQL (Zhong et al., 2017) and WTQ (Shi et al., 2020), which are free and open datasets for research use. All claims in this paper are based on the experimental results. Every experiment can be conducted on a single Tesla V100. No demographic or identity characteristics information is used in this paper.

## References

Philip Arthur, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Semantic parsing of ambiguous input through paraphrasing and verification. *Transactions of the Association for Computational Linguistics*, 3:571–584.

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 743–753, Melbourne, Australia. 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.

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the role of schema linking in text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6943–6954, Online. Association for Computational Linguistics.

Yuntao Li, Bei Chen, Qian Liu, Yan Gao, Jian-Guang Lou, Yan Zhang, and Dongmei Zhang. 2020. “what do you mean by that?” a parser-independent interactive approach for enhancing text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6913–6922, Online. Association for Computational Linguistics.

Jeremiah Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax Weiss, and Balaji Lakshminarayanan. 2020. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Advances in Neural Information Processing Systems*, 33:7498–7512.

Qian Liu, Dejian Yang, Jiahui Zhang, Jiaqi Guo, Bin Zhou, and Jian-Guang Lou. 2021. Awakening latent grounding from pretrained language models for semantic parsing. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1174–1189, Online. Association for Computational Linguistics.

Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. 2020. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Xinyu Pi, Bing Wang, Yan Gao, Jiaqi Guo, Zhoujun Li, and Jian-Guang Lou. 2022. Towards robustness of text-to-SQL models against natural and realistic adversarial table perturbation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2007–2022, Dublin, Ireland. Association for Computational Linguistics.

Ana-Maria Popescu, Oren Etzioni, and Henry A. Kautz. 2003. Towards a theory of natural language interfaces to databases. In *IUI ’03*, pages 100–112. IEEE.

Karthik Radhakrishnan, Arvind Srikantan, and Xi Victoria Lin. 2020. ColloQL: Robust text-to-SQL over search queries. In *Proceedings of the First Workshop on Interactive and Executable Semantic Parsing*, pages 34–45, Online. Association for Computational Linguistics.

Tianze Shi, Chen Zhao, Jordan Boyd-Graber, Hal Daumé III, and Lillian Lee. 2020. On the potential of lexico-logical alignments for semantic parsing to SQL queries. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1849–1864, Online. Association for Computational Linguistics.

Daniil Sorokin and Iryna Gurevych. 2018. Mixing context granularities for improved entity linking on question answering data across entity categories. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 65–75, New Orleans, Louisiana. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 173–179, Bergen, Norway. Association for Computational Linguistics.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. 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.

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

Zhiyong Wu, Ben Kao, Tien-Hsuan Wu, Pengcheng Yin, and Qun Liu. 2020. *PERQ: Predicting, Explaining, and Rectifying Failed Questions in KB-QA Systems*, page 663–671. Association for Computing Machinery, New York, NY, USA.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3879–3889, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ziyu Yao, Yu Su, Huan Sun, and Wen-tau Yih. 2019. Model-based interactive semantic parsing: A unified framework and a text-to-SQL case study. 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 5447–5458, 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.

Jichuan Zeng, Xi Victoria Lin, Steven C.H. Hoi, Richard Socher, Caiming Xiong, Michael Lyu, and Irwin King. 2020. Photon: A robust cross-domain text-to-SQL system. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 204–214, Online. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 1:135–154.

## A Implementation Details

### A.1 Baselines Implementation

We modified ETA since the vanilla version (Liu et al., 2021) does not support ambiguous and unanswerable span detection. For a fair comparison, we update the vanilla ETA in two ways. Firstly, in the original inference part, the vanilla version of ETA applies the greedy linking algorithm to only keep the top 1 confidence score-related schema item (column or value). We change the selection process by allowing the top 3 candidates to be chosen, whose confidence score should greater than the threshold. We consider those spans with multi-grounding results as ambiguous spans. Second, to enable ETA to handle unanswerable questions, we add a UNK column to the schema part and train the model to enable unanswerable span to get closer to the UNK column. We consider those linked with the UNK column as unanswerable spans. The heuristic-based baseline (Sorokin and Gurevych, 2018) is n-gram matching via enumerating all n-gram ($n \leq 5$) phrases in a natural language question and links them to schema items by fuzzy string matching. We consider a span as an ambiguous one when it can fuzzy match multiple results. Similarly, if a noun phrase span can match no results, it is considered to unanswerable span.

### A.2 DTE Implementation

Our DTE model consists of a BERT encoder, and three task modules, namely the concept prediction module, grounding module, and sequence labeling module. We implement the first two modules following the implementation details mentioned in Liu et al. (2021) and use the same hyperparameters. In addition, the sequence labeling module is built by a dropout layer, a linear layer, and a CRF layer which is based on the open-source repository pytorch-crf[3]. The response template for ambiguous questions is “Oops, this question has multiple semantic meanings. $X$ may refer to either "concept1", "concept2", or "$c_3$"”. What’s more, we design the template for unanswerable questions as “ Sorry, we can’t find an answer for you since "$X$" cannot be mapped to any concepts in your table”. Examples can be seen in Figure 1.

### A.3 Response Templates

The response template for ambiguous questions is “Oops, this question has multiple semantic meanings. $X$ may refer to either "concept1", "concept2", or "$c_3$"”. What’s more, we design the template for unanswerable questions as “ Sorry, we can’t find an answer for you since "$X$" cannot be mapped to any concepts in your table”. Examples can be seen in Figure 1.

### A.4 Training Hyper-parameters

For all experiments, we employ the AdamW optimizer and the default learning rate schedule strategy provided by Transformers library (Wolf et al., 2020). The learning rate of other non-BERT layers is $1 \times 10^{-4}$. The max training step is 100,000 and our training batch size is 35. The training process last 6 hours on a single 16GB Tesla V100 GPU.

## B Examples of NoisySP dataset

In this section, we demonstrate some good and bad cases of our DTE model prediction. Good case examples are shown in Table 8 and Table 9. Bad case examples are shown in Table 10 and Table 11.

| **Ambiguous Good Case of NoisySP** |
|---|
| **Q**: What is the minimum **population** of the parish with a 750.51 km area ?<br>**Gold:** O O O O **B-AMB** O O O O O B-VAL I-VAL I-VAL O<br>**Pred:** O O O O **B-AMB** O O O O O B-VAL I-VAL I-VAL O<br><br>**Schema**: Official Name \|\| Area km 2 \|\| foreign-born **population** \|\| total estimated **population**<br>**Description**: correct prediction |
| **Q**: Which **name** has a state of Yan ?<br>**Gold:** O **B-AMB** O O B-COL O B-VAL O<br>**Pred:** O **B-AMB** O O B-COL O B-VAL O<br><br>**Schema**: State \|\| Type \|\| born **name** \|\| first **name** \|\| Title \|\| Royal house \|\| From<br>**Description**: correct prediction |

Table 8: Ambiguous good cases by DTE on the NoisySP data.

| **Unanswerable Good Case of NoisySP** |
|---|
| **Q**: Which players college is Tennessee ?<br>**Gold:** O B-UNK B-COL O B-VAL O<br>**Pred:** O B-UNK B-COL O B-VAL O<br><br>**Schema**: Pick # \|\| NFL Team \|\| Position \|\| College<br>**Description**: correct prediction |
| **Q**: What prefecture is listed in the map as number 39 ?<br>**Gold:** O B-UNK O O O O B-COL O B-COL B-VAL O<br>**Pred:** O B-UNK O O O O B-COL O B-COL B-VAL O<br><br>**Schema**: Number in map \|\| Area (km²) \|\| Population (2001) \|\| Pop. density (/km²)<br>**Description**: correct prediction |

Table 9: Unanswerable good cases by DTE on the NoisySP data.

[3]https://github.com/kmkurn/pytorch-crf

**Ambiguous Bad Case of NoisySP**

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Q**: | I | want | the | date | of | appointment | for | manner | of | departure | being | sacked |
| **Gold:** | O | O | O | B-AMB | O | O | O | B-COL | I-COL | I-COL | O | B-VAL |
| **Pred:** | O | O | O | B-UNK | I-UNK | I-UNK | O | B-COL | I-COL | I-COL | O | B-VAL |

**Schema**: Team || Manner of departure || busy date || date of vacancy
**Description**: widened span error

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Q**: | What | is | the | winning | score | on | Feb | 12 | , | 1978 | ? |
| **Gold:** | O | O | O | O | B-AMB | O | B-VAL | I-VAL | I-VAL | I-VAL | O |
| **Pred:** | O | O | O | B-AMB | I-AMB | O | B-VAL | I-VAL | I-VAL | I-VAL | O |

**Schema**: home team score || score || Margin of Victory || Runner(s)-up || Date || Tournament
**Description**: widened span error

Table 10: Ambiguous bad cases by DTE on the NOISYSP data.

**Unanswerable Bad Case of NoisySP**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Q**: | How | many | hectars | of | land | is | in | Kaxholmen | ? |
| **Gold:** | O | O | B-UNK | I-UNK | I-UNK | O | O | B-VAL | O |
| **Pred:** | O | O | O | O | B-UNK | O | O | B-VAL | O |

**Schema**: Urban area (locality) || Municipality || Population || Density (inh./km²) || Code
**Description**: narrowed span error

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Q**: | What | was | the | elimination | number | of | the | fighter | who | fought | within | 26:15 | ? |
| **Gold:** | O | O | O | B-UNK | I-UNK | O | O | O | O | O | O | B-VAL | O |
| **Pred:** | O | O | O | O | O | O | O | O | O | O | O | B-VAL | O |

**Schema**: Wrestler || Entered || Eliminated by || Method of elimination || Time
**Description**: wrong detection error

Table 11: Unanswerable bad cases by DTE on the NOISYSP data.