Optimization Techniques for Unsupervised Complex Table Reasoning via Self-Training Framework

Zhenyu Li, Xiuxing Li, Sunqi Fan *Member, IEEE,* Jianyong Wang, *Fellow, IEEE*

Abstract—Structured tabular data is a fundamental data type in numerous fields, and the capacity to reason over tables is crucial for answering questions and validating hypotheses. However, constructing labeled data for complex reasoning tasks is labor-intensive, and the quantity of annotated data remains insufficient to support the intricate demands of real-world applications. To address the insufficient annotation challenge, we present a self-training framework for unsupervised complex tabular reasoning (UCTR-ST) by generating diverse synthetic data with complex logic. Specifically, UCTR-ST incorporates several essential techniques: we aggregate diverse programs and execute them on tables based on a "Program-Management" component, and we bridge the gap between programs and text with a powerful "Program-Transformation" module that generates natural language sentences with complex logic. Furthermore, we optimize the procedure using "Table-Text Manipulator" to handle joint table-text reasoning scenarios. The entire framework utilizes self-training techniques to leverage the unlabeled training data, which results in significant performance improvements when tested on real-world data. Experimental results demonstrate that UCTR-ST achieves above 90% of the supervised model performance on different tasks and domains, reducing the dependence on manual annotation. Additionally, our approach can serve as a data augmentation technique, significantly boosting the performance of supervised models in low-resourced domains[1].

Index Terms—Unsupervised Data Generation, Tabular Reasoning, Self Training

I. INTRODUCTION

TABULAR data is a widespread format for presenting information in the real world. This structure allows for the concise and efficient display of data. For instance, Wikipedia infoboxes utilize fixed-format tables to summarize relevant information with shared characteristics succinctly. Furthermore, tables are ubiquitous in various specific domains, such as scientific documents [1], financial reports [4], education [2], and industry [3]. Recent years have witnessed the remarkable development of tabular reasoning, which has achieved tremendous success in various downstream application areas. The fact verification task [5], [8] and the question-answering task [6], [7] are two prevalent reasoning tasks that assess a model's ability to comprehend and interpret tabular data effectively. Table fact verification is a natural language inference task [9] with evidence in structured forms.

Z. Li, S. Fan, and J. Wang are with the Department of Computer Science, Tsinghua University, Beijing 100084, China. E-mail: {zy-li21, fansq20}@mails.tsinghua.edu.cn, jianyong@tsinghua.edu.cn.

X. Li is with University of Chinese Academy of Sciences, Beijing, China. E-mail: lixiuxing@ict.ac.cn.

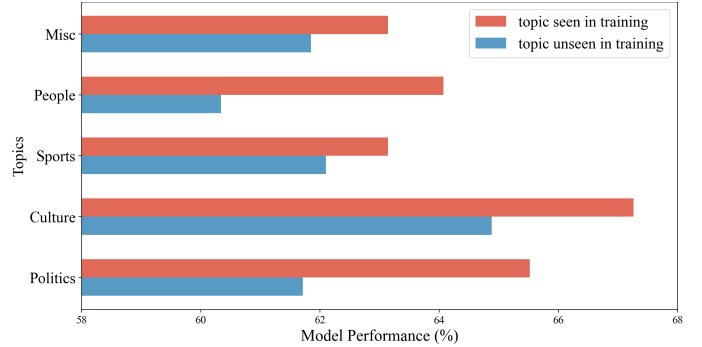[1]The code and models are available on https://github.com/leezythu/UCTR-ST.

Fig. 1.   The previous study [46] shows performance of models degrades dramatically on topics not seen during the training stage.

Given a table as evidence, the model is required to determine whether a textual hypothesis is "supported", "refuted", or "unknown". For table question answering, the model takes a table with a table-related natural language question as its input and returns the corresponding answer. Moreover, table-text reasoning tasks are introduced to better align with real-world requirements. The model must evaluate the table and related text concurrently to provide accurate judgments or answers. In summary, investigating tabular reasoning is crucial, as it facilitates more efficient techniques of the vast array of structured table resources that emerged on the web and in databases.

The essential challenges of tabular reasoning involve accurately comprehending table structures and effectively capturing the relationships between table cells or between table cells and related sentences. Despite the tremendous success of pre-trained language models (PLMs) in textual reasoning tasks (e.g., textual entailment [10], and question answering [11]), they primarily rely on free-form textual data for pretraining. Consequently, the considerable format discrepancy between free-form texts and structured tables seriously restricts effective table reasoning. To further alleviate the challenges that existed in tabular reasoning, an ever-growing tendency of adapting pre-trained models to tables has emerged [12], [13], [14], [15]. They explore diverse table-oriented model architectures [16], [17], [18] and pre-training objectives [19], [20], [21] to leverage the particular properties of the table better. Moreover, they investigate distinct serialization methods [26], [27] to linearize tables to sequences, attempting to eliminate the gap. These methods have achieved significant

Fig. 2. The comparison of simple claims and complex claims. A simple claim only involves a specific table cell, but a complex claim requires the annotator to consider the relationship among multiple cells.

improvements over previous approaches [22], [23], [24].

However, the aforementioned methods are based on the assumption that adequate training data is available, which may not always be true. When human-annotated data is insufficient, these approaches might suffer considerable limitations and experience substantial performance deterioration. Additionally, Chemmengath et al. [46] observe a significant decrease in the model's performance when encountering samples from topics not covered during the training stage. As demonstrated in Figure 1, the model's performance declines markedly when exposed to previously unseen topics. To emancipate the limitations of the above assumptions and make the setting of tabular reasoning more in conformity with the actual scenarios, the unsupervised complex tabular reasoning setting has been proposed, which means reasoning on tables or a hybrid of tables and related text using complex logic with no manually annotated data available. These methods can be generally divided into two optimization directions: (1) Methods based on pre-training process reconstruction. These methods are designed as data-augmentation techniques with limited unsupervised performance. In addition, they always require a large pre-training corpus. Yu et al. [19] pre-train the model on a large amount of question-SQL pairs, and Liu et al. [20] show that synthetic SQL queries can provide a better model initialization. (2) Methods based on synthesizing human-like data through heuristics or data-to-text models. Eisenschlos et al. [29] generate claims using context-free grammar (CFG) templates and counterfactual heuristics. Recently, Pan et al. [30] propose an unsupervised learning framework named MQA-QG. MQA-QG generates multi-hop questions for both the tabular and textual data, which is the most relevant work in this tendency.

Nevertheless, several critical issues remain unsolved in the realm of unsupervised complex tabular reasoning: (1) Existing methods for data generation mainly use heuristics or shallow data-to-text methods (e.g., converting a row to a sentence). Thus, they can merely generate relatively simple instances shown in Figure 2, limiting the model's effectiveness on complex reasoning samples, which require a deep understanding of the semantics and logic relationships between multiple table cells. (2) Previous works only focus on a single scenario but cannot be expanded to other tabular reasoning tasks.

This is because they design heuristics based on specific data characteristics or the form of the task, and these methods cannot be transferred to other tasks flexibly. Therefore, these models struggle to handle the complex and diverse scenarios encountered in the real world.

To address these issues, we introduce UCTR-ST (**U**nsupervised **C**omplex **T**abular **R**easoning using **S**elf-**T**raining), an advanced self-training framework designed explicitly for unsupervised complex tabular reasoning. More specifically, UCTR-ST primarily leverages a random sampling strategy to collect different types of programs. These programs consist of sequences of symbols that can be executed on tables, including SQL queries, logical forms, and arithmetic expressions, encompassing a wide range of reasoning types. Subsequently, we design a "Program-Management" module that generates program-answer pairs by leveraging numerous tables within the domain. To bridge the gap between the programs and natural language sentences, we develop a powerful "Program-Transformation" module based on generative language models that turns the programs into human-like natural questions or claims with complex logic. Since a table often occurs with its surrounding texts, UCTR-ST also defines a "Table-Text Manipulator". It contains two basic operators: Table-To-Text and Text-To-Table, to fuse information from table and text sources. Based on the combination of these components, UCTR-ST can handle question-answering and fact-verification tasks under both the homogeneous (table only) setting and the heterogeneous (hybrid table and text) setting, aiming for a unified framework. Figure 3 illustrates the progress that UCTR-ST generates a joint table-text reasoning sample through a SQL query, utilizing our foundational modules and operators. Experiment results show that UCTR-ST can generate diverse and human-like training samples with complex logic, which results in surprising unsupervised performance. Additionally, existing models pre-trained on our synthetic dataset significantly outperform the supervised model under the few-shot setting. Finally, we reconstruct the model's training process via the self-training framework, refraining from an exclusive reliance on synthetic data, which may not accurately represent the distribution of real-world data. Therefore, UCTR-ST can further improve the model's performance by effectively leveraging unlabeled realistic data. UCTR-ST employs the model initially trained on synthetic data to assign labels to the unlabeled real-world data, incorporating these pseudo-labeled samples into the training set. By repeating this process, the model achieves self-boosting, resulting in significant performance improvements. The experimental results indicate that UCTR-ST effectively utilizes unlabeled data to achieve remarkable enhancements compared to UCTR, a basic version of UCTR-ST that doesn't use the self-training technique. Notably, we also discover that UCTR-ST can considerably enhance fully-supervised performance in low-resource domains.

Our main contributions can be summarized as follows:

- To the best of our knowledge, this is the first study exploring a unified unsupervised complex tabular reasoning framework.

- We propose a novel and effective framework by leveraging program generation and conversion modules to cope with unsupervised complex reasoning.
- We further design novel "Table-to-Text" and "Text-to-Table" operators in UCTR to handle joint table-text reasoning scenarios.
- In order to better utilize the information from the distribution of unlabeled real data, we employ self-training techniques to enable the model to self-boost, which demonstrates general effectiveness of UCTR-ST across various tasks.
- Comprehensive experiments show that UCTR and UCTR-ST can significantly benefit tabular reasoning systems under unsupervised, few-shot, and even supervised settings.

II. PRELIMINARY

In this section, we start with introducing the background knowledge of the tabular reasoning task. In particular, we first present related basic concepts and then formalize the tabular reasoning task. Afterwards, since this paper aims to tackle the unsupervised scenario, we give a brief overview of the primary unsupervised data generation approaches.

A. Tabular Reasoning.

We first define some basic concepts related to the tabular reasoning task.

Table. The structure of a table can be very flexible, and we can divide tables into various categories according to their different formats [31]. Among them, relational tables are the most commonly used. For a relational table T with n rows $\{r_1, ..., r_n\}$, each row can be seen as a record, with columns as the corresponding attributes.

Context. In most cases, there are related paragraphs P surrounding a table as its context. These texts always describe the table's contents or contain supplementary information. Some tabular reasoning tasks require the model to consider not only the evidence from tables, but also the evidence in textual form. Reasoning on heterogeneous data is more realistic and challenging.

Tabular Reasoning. In this paper, we define tabular reasoning as reasoning tasks on tabular evidence or joint table-text evidence. Specifically, we use two tasks: tabular fact verification and tabular question answering, to evaluate a model's reasoning ability. We can formalize the task as a mapping from the evidence and a natural language sentence L to an output O. The basic mapping can be written as:

$$f(T, L) \rightarrow O \tag{1}$$

If the evidence consists of both a table and its related text, the mapping can be extended as:

$$f(T, P, L) \rightarrow O \tag{2}$$

We present detailed explanations of the equation for each specific task below.

Tabular Fact Verification. Given a table as evidence, tabular fact verification requires the model to judge whether the evidence supports, refutes a natural language claim, or it's

unknown. That is, $O \in \{Supported, Refuted, Unknown\}$ in Equation 1 and 2. It is similar to traditional fact verification task on textual data [32], except the evidence format.

Tabular Question Answering. Similarly, tabular question answering is a migration of the traditional question answering task from textual data to tabular data. Its output is always a specific answer inferred from the evidence.

Complex tabular reasoning. We define complex tabular reasoning as the reasoning process of considering multiple table cells and understanding their logical relationships to infer the correct answer. In contrast, simple tabular reasoning only involves a single table cell, as depicted in Figure 2. Simple reasoning tasks are easier to solve, since models are good at learning associations between surface texts.

Program. A program is an executable sequence of symbols [34], such as a SQL query. Unlike natural language texts, programs have strict grammar rules with no ambiguity and have definite execution results. As a related concept, "program context" refers to an environment where a program is applied. The variables used in the program are also sampled from the context. For example, tables are the corresponding context for SQL queries. Besides, we refer to a "program executor" as an automated tool that executes a program within the context, such as a SQL executor. We can use the program executor as a black box, whose input is a program and program context, and output is the execution result.

Section II-C elaborates on the types of programs we used in this paper.

B. Unsupervised Data Generation.

Supervised models tend to show powerful results in an ideal environment where sufficient high-quality data is available. Unfortunately, we often face situations with a limited amount of labeled data or no labeled data in the real world, under which the model's performance suffers a severe decline inevitably. This dilemma leads to the research direction of unsupervised data generation, aiming to synthesize human-like training instances [33].

Formally, for tabular reasoning tasks, supervised models assume labeled training data $X = \{(t_1, p_1, l_1, o_1), \cdots, (t_i, p_i, l_i, o_i), \cdots, (t_n, p_n, l_n, o_n)\}$, where n is the number of training instances. But under unsupervised settings, we only have $X = \{(t_1, p_1), \cdots, (t_i, p_i), \cdots, (t_n, p_n)\}$ as available information, where t_i, p_i, l_i, and o_i are an unlabeled table, the related text, a natural language question/claim, and the corresponding golden label, respectively. The data generation method tries to reconstruct a synthetic training dataset $X^{'} = \{(t_1, p_1, l_1^{'}, o_1^{'}), \cdots, (t_i, p_i, l_i^{'}, o_i^{'}), \cdots, (t_n, p_n, l_n^{'}, o_n^{'})\}$ using these raw tables and texts. Based on this synthetic dataset, supervised models can be applied successfully.

However, the distribution of the generated data in the above manner may have a significant gap from the distribution of questions/claims from real users. Therefore, we can adopt a relaxed but more practical unsupervised data generation setting: we have $X = \{(t_1, p_1, l_1), \cdots, (t_i, p_i, l_i), \cdots, (t_n, p_n, l_n)\}$ as available information. In the subsequent experiments, we

demonstrate that, guided by the information of real questions/claims, the model can achieve better performance on real test data.

C. Program Design.

In this paper, we adopt three types of programs: logical forms, SQL queries and arithmetic expressions. We depict examples of their forms and execution results on a table in Figure 4. Among them, logical forms are used for fact verification tasks, while SQL queries and arithmetic expressions are used for question answering tasks. Due to the variety of logic operators and flexible structure, the programs can cover most types of logic used in tabular scenarios. We give more detailed explanation of each program type below:

SQL Queries. SQL is standard language for managing data, which is widely used in relational databases. SQL supports many manipulations like query, insert, update, and delete, but we only need SQL queries for our reasoning setting. In most cases, you can query any content you want to know from the table through one or more SQL queries. Specifically, the SQL queries support the following reasoning types (conditions): $equivalence$ $(=)$, $comparison$ $(>, <, order\ by, max, min)$, $counting$ $(count)$, sum $(+)$, $diff$ $(-)$ and $conjunction$ (and).

Logical Forms. Though SQL queries are powerful, they cannot be directly used on tabular fact verification tasks. So in our framework, we generate factual claims based on logical forms specifically. A Logical form is a symbolic formulation that can be executed on database tables to judge the truthfulness of the inner logic. Logical Forms can also support most common reasoning types such as: $count$, $superlative$, $comparative$, $aggregation$, $majority$, $unique$, and $ordinal$. For example, in the logical form depicted in Figure 4, $argmax$ returns the row with the max value under the specified column, and hop extracts the value under a specified column for an input row. Finally, eq judges whether the two arguments are equal. Due to the space limitation, we refer readers to [35] for a complete list of the operations. Due to the limited space, we refer readers to [35] for the full list of operators.

Arithmetic Expressions. Arithmetic expressions can be used to express complex arithmetic operations. As shown in Figure 4, an arithmetic expression consists of a sequence of operations. Arithmetic expressions support 6 mathematical operations: add, $subtract$, $multiply$, $divide$, $greater$, exp and 4 table aggregation operations $table\_max$, $table\_min$, $table\_sum$, $table\_average$. We refer readers to [28] for more detailed illustrations.

III. FRAMEWORK

In this section, we present our proposed self-training framework for unsupervised complex tabular reasoning, UCTR-ST. UCTR-ST uses three essential modules to generate human-like data for various tabular reasoning scenarios: Program-Management, Program-Transformation, and Table-Text Manipulator. The left part of Figure 3 shows how UCTR-ST generates joint table-text reasoning instances using SQL queries

on a table from the TAT-QA dataset. Note that the Table-Text Manipulator consists of two operators: Table-To-Text and Table-To-Text, corresponding to two different data generation methods. The right part of the figure depicts the self-training process. We show more details of each part of the workflow below.

A. Table Splitting.

As shown in the left part of Figure 3, given a raw table, the table splitting generation method first executes a program based on the Program-Management module and gets an answer. Note that not all table cells affect the final output, and we define the cells involving the reasoning process as "highlighted cells." Then the Table-To-Text operator selects one highlighted cell, and transforms the row where the cell is located into a natural language text, keeping the rest of the rows as a sub-table. Additionally, the Program-Transformation turns the program into a question with the same meaning. In this way, we successfully synthesize a training instance $(t, p, l) \rightarrow o$ requiring evidence from both a table and its related text.

B. Table Expansion.

Table expansion can be regarded as an inverse process of table splitting. The table splitting method synthesizes joint table-text reasoning instances from only tables, while the table expansion method tries to integrate information from the original texts surrounding the table. Specifically, the table expansion method first finds the relevant sentences and then uses the Text-To-Table operator to transform essential information of the sentences into tabular form. If the generated table shares the same row name or the same column name with the original table, they can be integrated into a new expanded table. Afterwards, UCTR-ST can apply the Program-Management and Program-Transformation techniques on this expanded table as in the table splitting method. Finally, we synthesize a joint table-text reasoning instance with evidence from the original table and text.

Table-To-Text and Text-To-Table operators are designed for joint reasoning on heterogeneous data. For table-only scenarios, we can follow the same procedure but just use the Program-Management and Program-Transformation modules. Thus, UCTR-ST can become a unified framework that can cope with both homogeneous and heterogeneous scenarios.

C. Tabular Reasoning Models.

Although researchers have designed different model structures for various tasks, the mainstream methods share the same paradigm as follows:

$$e_i = Encoder(t_i, p_i, l_i)$$
$$\theta_{model} = \arg\min_{\phi} L(Classifier(e_i), o_i) \quad (3)$$

where t_i, p_i, and l_i represent the table, paragraph and natural language sentence in a sample. o_i is the golden label of the sample. L is the loss function, and θ is the model parameters. We first get a joint representation based on an encoder like BERT [40], then a designed classifier is applied
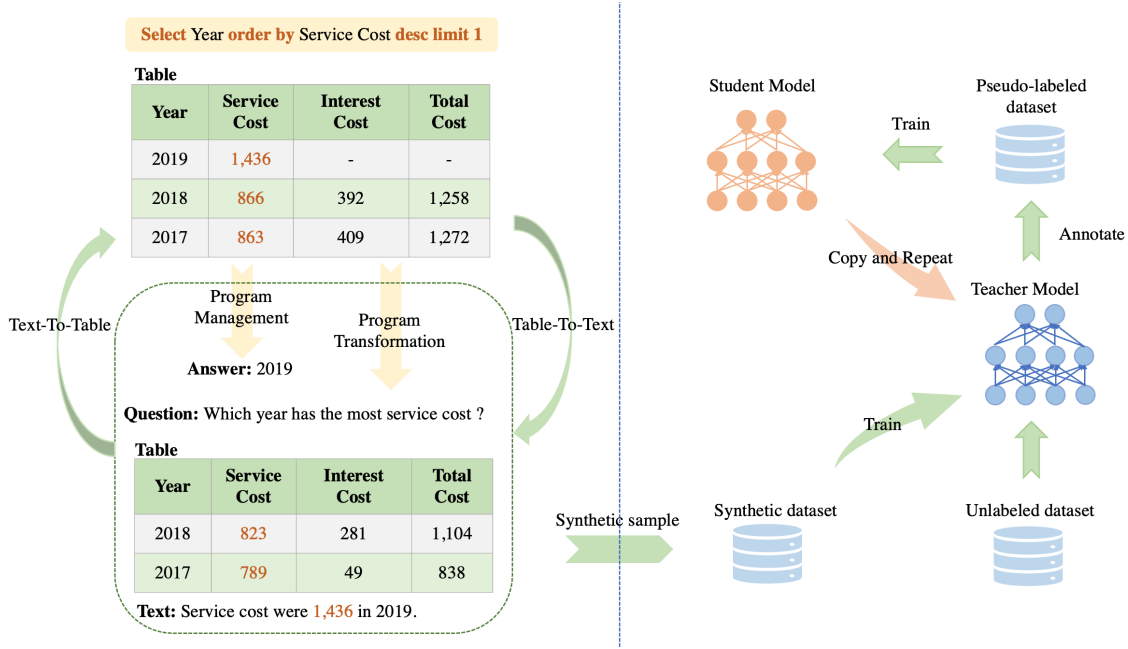
Fig. 3. Illustration of our framework. The left part depicts how we generate synthetic samples (enclosed in the dashed box). Specifically, the Table-To-Text operator focuses on splitting the original table into a sub-table and a generated sentence and then building a joint table-text reasoning sample based on the basic modules. The Text-To-Table operator adopts a similar procedure but aggregates information from the original table and text to form an expanded table. The right part show how we apply the self-training technique. In each iteration, the teacher model infers on the unlabeled data to generate pseudo-labeled data, which is then used to train a better student model.

to the representation to get predicting results. We optimize the model's parameters using gradient descent techniques during training. In experiments, we use the representative model on each task as the supervised baseline.

D. Self-Training

Self-training is a popular technique in semi-supervised machine learning, which involves training a model with a small set of labeled data and a larger set of unlabeled data. But there are relatively few works that use the self-training technique in completely unsupervised scenarios. In this paper, we recognize synthetic samples as the existing labeled data and the samples in the training set as the unlabeled data that conforms to the natural distribution.

Specifically, we train an initial model based on the synthetic data and then make predictions on the unlabeled dataset. The examples for which the model makes predictions are then added to the labeled dataset and used to re-train the model. This iterative process can improve the robustness of the model and also the quality of the labeled dataset with each iteration. Eventually, the model can converge to a better performance.

IV. METHODOLOGY

In this section, we present the workflow of the UCTR-ST framework, along with more detailed information on each technique used, demonstrating how they can achieve our ultimate goals: i) generating human-like training instances with complex logic, ii) being able to handle various table reasoning scenarios, and iii) achieving better testing performance with unlabeled real-world data. Specifically, we first formalize the

modules and their corresponding functions included in each technique, and then provide specific explanations for how we apply programs and training models.

A. Program-Management.

This component aims to aggregate diverse programs with complex logic and execute them on tables. Formally, we can define these two procedures as follows:

Program Generation. Given raw tables in a specific domain, this procedure retrieves program templates from a template pool and applies these templates to tables to get executable programs:

$$f(T) \rightarrow Prog \tag{4}$$

Program Execution. The function of our Program Execution component is the same as stated in the preliminary. Given a table and a program as input, the executor returns the execution result:

$$f(T, Prog) \rightarrow O \tag{5}$$

Programs in each type rely on a specific executor. We give a more detailed explanation for these programs in section IV-D and section V-B.

B. Program-Transformation

The advantage of programs compared to natural language is that there is no ambiguity and they can give definite execution results according to the grammar rules so that we can get concise program-answer pairs. However, different types of programs follow different grammar rules, and there

is a huge gap in the surface form between a program and a natural language sentence. Therefore, we design a program-transformation component for mapping different programs of different types into a unified natural language format. Formally, it can be regarded as a mapping function as follow:

$$f(P) \rightarrow L \tag{6}$$

where P is a program and L is the corresponding natural language sentence with the same meaning.

C. Table-Text Manipulator

Table-To-Text. This operator converts a table into a sub-table and a generated sentence. Formally, the function is defined as:

$$f(T) \rightarrow T_{sub}, S \tag{7}$$

Specifically, we follow the implementation of "DescribeEnt" operator in [30] to transform a row into a natural language sentence, and more advanced models [36], [37] can also be used here. Additionally, we add a filtering step. That is, if important information in the table is missing from the generated sentence, we will discard it.

Text-To-Table. As an inverse process of Table-To-Text, the function of Text-To-Table can be written as:

$$f(T, P) \rightarrow T_{expand} \tag{8}$$

Actually, text-to-table is a recently proposed task for information extraction [38]. But current techniques do not support integrating text information into existing tables. So a filtering step is also needed here. We first use row names to filter possible useful sentences, and then apply a text-to-table model proposed in [38] to get a generated table with only one record. Finally, we integrate this record into the original table to form an expanded table.

D. How we collect program templates

The three types of programs (SQL queries, logical forms, and arithmetic expressions) are essential parts of UCTR-ST. In this section, we explain the necessity of each type of program and show how we collect templates and apply them on tables to get program-answer pairs. We depict examples of each type of program in Figure 4.

For SQL query templates collection, we follow the implementation in [20], using templates extracted from SQUALL [39]. SQUALL is a dataset consisting of question-SQL pairs with manual alignments. One example template from SQUALL is as follows:

$$select \ \textbf{c1} \ from \ \textbf{w} \ order \ by \ \textbf{c2\_number} \ desc \ limit \ 1$$

where w represents the table, and $c1$ and $c2$ correspond to the first and the second column. $\_number$ indicates that the data of this column is numerical. These placeholders allow the template to migrate to other tables conveniently.

For logical forms, we use the LOGIC2TEXT dataset proposed by Chen et al. [35]. LOGIC2TEXT consists of a large number of claim-program pairs, covering most common logic types such as count, comparative, aggregation etc. We directly

sample program templates from it. Here is an example of the template:

$$eq \ \{ \ hop \ \{ \ filter\_eq \ \{ \ all\_rows \ ; \ \textbf{c1} \ ; \ \textbf{val1} \ \} \ ; \ \textbf{c2} \}; \textbf{val2} \}$$

where $val1$ and $val2$ are cell values from the first and second columns. eq, hop, and $filter\_eq$ are defined operations. Specifically, $filter\_eq$ returns rows that satisfy the constraints.

Arithmetic operations are very common in some specific tabular reasoning scenarios (like financial and scientific). Although SQL can implement most types of operations, expressing arithmetic operations using SQL always results in very long sequences. Thus, we adopt arithmetic expressions for tabular reasoning tasks involving arithmetic operations as our programs. Specifically, we collect templates of arithmetic expressions from the Finqa dataset proposed in [28]. The original form of a template is as follows:

$$subtract(\textbf{val1}, \textbf{val2}), divide(\textbf{\#0}, \textbf{val2})$$

where $\#0$ denotes the result from the first $subtract$ step. But the original form doesn't contain the information of the row's name or column's name, so we further replace $vali$ with $col\_name \ of \ row\_name$, where col_name and row_name are the column's name and row's name corresponding to $vali$. For more details about the arithmetic operations please refer to [28].

E. How to apply program templates

We call the column names, and cell values involved in the program template as column-placeholders and value-placeholders, respectively. To apply these programs to a new table, we need to fill these placeholders with variables from the table. Here we adopt the random sampling strategy with type constraints for program sampling. Specifically, we first populate the column-placeholders by randomly sampling from the columns of the new table. Afterwards, for each column, we randomly sample the values in it to populate the value-placeholders. Besides, if the column-placeholder specifies a data type (e.g., number, string), we only sample from columns that match that type.

Take the logical form above as an example. The original program template is:

$$eq \ \{ \ hop \ \{ \ filter\_eq \ \{ \ all\_rows \ ; \ \textbf{c1} \ ; \ \textbf{val1} \ \} \ ; \ \textbf{c2} \}; \textbf{val2} \}$$

For a new table T, we first fill in the column-placeholders:

$$\{c1, c2\} \leftarrow Random\_Sample(T.columns, data\_type)$$

Then we fill in each value-placeholders:

$$val1 \leftarrow Random\_Sample(c1.values)$$
$$val2 \leftarrow Random\_Sample(c2.values)$$

In practice, for logical form templates with a format $func \ \{ \ arg1 \ ; \ arg2 \ \}$, in which $func$ is the root operator, $arg1$ is a complex sub-template, and $arg2$ is a single value. We first apply sampling on $arg1$ and execute it. Then we can determine the value of $arg2$ based on the execution result and the root operator to obtain a true/false claim.

In summary, this mapping strategy keeps the internal relationship of the variables in the original program. Moreover,
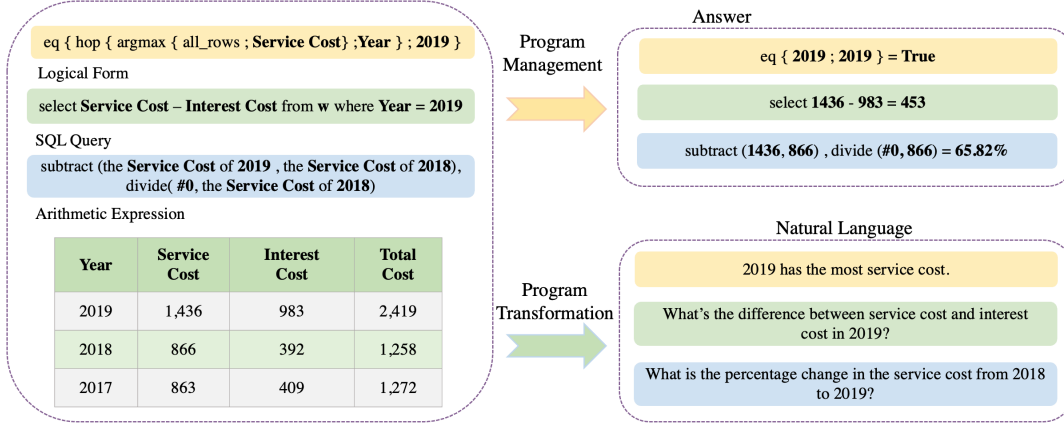
Fig. 4. Examples of three types of programs we used in this work: logical forms, SQL queries, and arithmetic expressions. The Program-Transformation module transforms a logical form into a claim and transforms the other two types of programs into questions.

this strategy is naturally suitable for evidence-based reasoning tasks since the values sampled during the process are exactly the evidence associated with the synthetic instance. Notably, if the execution result is empty, we discard this program.

Algorithm 1 Data Generation Procedure

Require: Table-text dataset D_t, program template dataset D_m
Ensure: Target dataset D consisting of (L, T, P, O) pairs
1: $D \leftarrow []$
2: **for** table, paragraph in D_t **do**
3:     $table^{exp} \leftarrow \text{TextToTable}(table, paragraph)$
4:     **for** template in D_m **do**
5:         $prog \leftarrow \text{Sampling}(table, template)$
6:         $prog^{exp} \leftarrow \text{Sampling}(table^{exp}, template)$
7:         $ans \leftarrow \text{Executor}(table, prog)$
8:         $ans^{exp} \leftarrow \text{Executor}(table^{exp}, prog^{exp})$
9:         **if** ans or ans^{exp} is empty **then**
10:             **continue**
11:         **end if**
12:         $table^{sub}, sentence \leftarrow \text{TableToText}(table)$
13:         $NL \leftarrow \text{Transformation}(prog)$
14:         $NL^{exp} \leftarrow \text{Transformation}(prog^{exp})$
15:         $D_t.\text{append}((NL, table^{sub}, sentence, ans))$
16:         $D_t.\text{append}((NL^{exp}, table, paragraph, ans^{exp}))$
17:     **end for**
18: **end for**

F. How to train the generative model.

For the program-transformation, there are few works on converting a program to a natural language sentence. This paper tackles this problem based on generative language models. For logical forms, we directly use the fine-tuned GPT-2 [42] model on the Logic2Text [35] dataset. For SQL queries and arithmetic expressions, we fine-tune a BART [41] model ourselves on SQUALL [39] and Finqa [28], respectively. These

---

Algorithm 2 Self Training Algorithm

Require: Synthetic dataset $D_s = \{(l_i^s, t_i^s, p_i^s, o_i^s)\}$ generated from Algorithm 1 as labeled data; Unlabeled realistic dataset $D_r = \{(l_i^r, t_i^r)\}$.
Ensure: Target model parameter θ_{final}
    Step 1. Fine-tune teacher model θ_{tea} on labeled synthetic data
1: $\theta_{\text{tea}} = \arg\min_\theta \text{ loss}\{(l_i^s, t_i^s, p_i^s, o_i^s)\}$
2: **while** not converged **do**
    Step 2. Generate pseudo-label for unlabeled real dataset
3:     $(p_i^r, o_i^r) \leftarrow f((l_i^r, t_i^r); \theta_{\text{tea}})$
    Step 3. Merge the labeled synthetic data and pseudo-labeled realistic data into a union of data
4:     $D_u = \{(l_i^s, t_i^s, p_i^s, o_i^s)\} \cup \{(l_i^r, t_i^r, p_i^r, o_i^r)\}$
    Step 4. Fine-tune student model θ_{stu} on the union of labeled synthetic data and pseudo-labeled real data
5:     $\theta_{\text{stu}} = \arg\min_\theta \text{ loss}\{(l_i^u, t_i^u, p_i^u, o_i^u)\}$
    Step 5. Update the teacher model
6:     $\theta_{\text{tea}} \leftarrow \theta_{\text{stu}}$
7: **end while**

---

three datasets contain program-NL training pairs for each type of program. Here we also briefly introduce generative models (e.g., GPT-2 and BART). They are transformer-based models pre-trained on a large corpus of text in an unsupervised manner and have been demonstrated to be very effective on machine translation tasks. For more details please refer to [42], [41]. In our work, we recognize converting a program to a natural language sentence as a translation task, that is, translating a program into a sentence. Specifically, we fine-tune generative models in an end-to-end manner:

$$L = Generative\_Models(Prog) \tag{9}$$

Here the generative models can be GPT-2, T5 and BART, etc.

In summary, we first collect a set of diverse program templates and then apply them to new tables by random

TABLE I
DATASET STATISTICS OF FEVEROUS, TAT-QA, WiKiSQL AND SEM-TAB-FACTS.

Dataset	Domain	Total Samples	Evidence Type	Label/Question Types
FEVEROUS	Wikipedia	87,026	34,963 sentences, 28,760 tables 24,667 combined	49,115 Supported, 33,669 Refuted 4,242 NEI
TAT-QA	Finance	16,552	7,431 tables, 3,902 sentences 5,219 combined	9,211 Span/Spans, 377 Counting 6,964 Arithmetic
WIKISQL	Wikipedia	80,654	24,241 tables	43,447 What, 5,991 How many 5,829 Who, ...
SEM-TAB-FACTS	Science	5,715	1,085 tables	3,342 Supported, 2,149 Refuted 224 Unknown

sampling variables from the new context to get valid programs. Finally, we convert these programs into human-like reasoning instances using four basic components.

We depict the overall data synthesizing procedure using both table splitting method and table expansion method in Algorithm 1, and the self-training procedure in Algorithm 2.

V. PERFORMANCE EVALUATION

A. Dataset and Evaluation.

Datasets. To test the effectiveness of our UCTR-ST framework, we apply it in various settings. We conduct extensive experiments on four representative benchmarks: FEVEROUS [45], TAT-QA [4], WiKiSQL [6], and SEM-TAB-FACTS [1]. The four datasets cover fact verification and question answering tasks under table-only and table-text reasoning scenarios, in general and specific domains. Here we give a brief introduction to each dataset. FEVEROUS is a dataset for fact verification over evidence from sentences and tables within Wikipedia. TAT-QA is also built on hybrid data but aims for question answering task. Additionally, its evidence is extracted from real-world financial reports. WiKiSQL consists of examples of questions and SQL queries over tables from Wikipedia. SEM-TAB-FACTS is a fact verification dataset with evidence in tabular form, and the tables are from scientific articles. Statistics of these datasets are shown in Table I.

Evaluation protocol. There are different evaluation metrics for different benchmarks. Typically, the pipeline of a model on FEVEROUS consist of retrieving stage and reasoning stage. In the first stage, the model retrieves sentences and table cells related to a claim from Wikipedia. Then in the second stage, the model judges whether the claim is supported, refuted, or there is not enough information (NEI) based on the evidence. So for FEVEROUS, the metrics are label accuracy and FEVEROUS score. Label accuracy measures the proportion of the number of correct labels predicted by the model to the total number. FEVEROUS score is a more strict metric that considers both the retrieving stage and the reasoning stage. For a sample, only when both the retrieved evidence set and the predicted label are correct is the prediction considered correct. Since the retrieving stage is not the focus in our paper, we directly use the retriever proposed in [45] as our first-stage model and only experiment with the reasoning stage. Notably, we train

the reasoning model on the golden evidence set rather than the retrieved evidence set, as the latter contains much noise. The metrics to measure performance on TAT-QA are Exact Match (EM) and numeracy-focused F1 score [43]. For WiKiSQL, the evaluation metric is denotation accuracy, which measures how many predicted answers are equal to the ground-truth answers. For SEM-TAB-FACTS, we adopt the standard 3-way micro F1 from the original paper. This metric evaluates whether claims are classified as Supported, Refuted, or Unknown.

B. Implementation Details.

For the "Program-Transformation" module, we choose the appropriate program type according to the setting of a task and the reasoning ability it requires. Specifically, we apply logical forms on FEVEROUS and SEM-TAB-FACTS tasks during the data generation procedure to generate claims with complex logic, and apply SQL queries on WiKiSQL. For TAT-QA, we apply both SQL queries and arithmetic expressions. As shown in Table I, there are various reasoning types in TAT-QA. We use SQL queries to handle the Span/Spans type and use arithmetic expressions for the Counting and Arithmetic type. The tables we use to generate synthetic data are from the original datasets. Finally, we get 79,856, 23,933, 27,365, 4,071 synthetic samples for FEVEROUS, TAT-QA, WiKiSQL and SEM-TAB-FACTS, respectively.

As for model training, the entire process is based on a self-training framework. After each iteration, we either add the generated pseudo-labeled data to the original synthetic dataset or only use the generated pseudo-labeled data, depending on the performances on the dev set. We select representative models on the four benchmarks as our baselines. Specifically, we adopt the models in the original papers of FEVEROUS [45] and TAT-QA [4]. Since they achieve good results with reproducible codes. For WiKiSQL, we use the current state-of-the-art model TAPEX [20]. For SEM-TAB-FACTS, we use a representative model, TAPAS [13]. Section V-C shows more details of the models used on each benchmark. In experiments, we follow the implementation in [55] on FEVEROUS, only predicting the "Supported" or "Refuted" label, since the "NEI" label occupies a tiny proportion of the dataset. Besides, we also evaluate models under a few-shot setting, where we assume

only 50 human-labeled samples are available. The 50 samples are randomly selected from the original training set.

The executor for SQL queries is sqlite3 [2]. For logical forms and arithmetic expressions, we utilize the executor proposed in [35] and [28], respectively. Experiments are conducted with 4 GeForce RTX 3090 graphics cards.

C. Results Analysis.

Table II, III, and V summarize the unsupervised and few-shot results on three benchmarks. In this section, we analyze the effectiveness of our self-training framework for unsupervised complex tabular reasoning (UCTR-ST) compared to supervised baselines. We first give illustrations of the supervised and unsupervised models we use. Representative supervised models are as follows:

(1) TAGOP is a strong supervised model designed for TAT-QA. It first tags relevant table cells and text spans and then reasons over these elements using a set of predefined operators. Text-Span only and Table-Cell only are two weak supervised baselines that adopt the same architecture as TAGOP, but they focus on textual evidence or tabular evidence only.

(2) Full baseline is the baseline model proposed in [45] that consists of a retriever module that retrieves relevant table cells and sentences from Wikipedia and a verdict predictor that predicts a label. As mentioned above, since we only focus on the reasoning stage, we assume golden evidence is available when testing label accuracy. When testing the FEVEROUS score, we use the trained retriever in the original paper for a fair comparison. The Sentence-only baseline and Table-only baseline are two weak supervised models trained only on sentences or tables.

(3) TAPAS is a popular tabular reasoning model, using joint pre-training of textual and tabular data. It uses special positional embeddings to encode table structures and shows promising performance on fact verification and question answering tasks. We apply TAPAS on both TAT-QA and SEM-TAB-FACTS. The result on TAT-QA is from [4]. For SEM-TAB-FACTS, we follow the method in [44] to fine-tune TAPAS.

(4) TAPEX is a generative pre-trained model that is pre-trained on a large SQL query-answer corpus to imitate a neural SQL executor, and it produces state-of-the-art results on the WiKiSQL dataset. We also use it as an unsupervised model to see how much the synthetic corpus can help the model cope with real questions. We evaluate the officially released tapex-base models and get the corresponding results on development and test sets.

We compare our UCTR-ST framework with the following unsupervised models:

(1) Random is a naive baseline used for FEVEROUS and SEM-TAB-FACTS, selecting a label randomly. Since these two tasks are essentially multi-classification tasks, this baseline shows how much performance a model should at least achieve. Notably, the "NEI" label in FEVEROUS only occupies a tiny proportion, so we only predict the "Supported" or "Refuted" label in practice.

(2) MQA-QG is also an unsupervised data generation method, which is the most relevant work to ours. Though it is initially designed for multi-hop question generation, we make some modifications to fit it on these benchmarks. Specifically, MQA-QG finds a bridge entity that connects the table and related text, then turns the row containing the bridge entity into a describing sentence using a $DescribeEnt$ operator. Finally, it aggregates the information from the describing sentence and the related text to form a question or a claim. MQA-QG can generate data from tables or a hybrid of tables and texts. But the main deficiency is that it cannot integrate the information from multiple rows using complex underlying logic, so the generated questions/claims are relatively simple.

(3) UCTR is a basic version of UCTR-ST, lacking the self-training process while keeping other components the same, and it cannot leverage unlabeled real data.

(4) UCTR $-w/o$ T2T is an ablation model of UCTR. It represents the UCTR framework without the Table-To-Text and Text-To-Table operators, so it cannot generate samples containing both tabular and textual information as evidence.

(5) TAPAS-Transfer is a transfer learning model from TABFACT [5]. TABFACT is a large dataset focusing on fact verification on Wikipedia tables. It consists of 117,854 human-annotated claims on 16,573 tables. This model is trained on TABFACT and then directly applied on SEM-TAB-FACTS.

According to the results shown in Table II, III, and V, we have the following observations:

(1) The basic version of our proposed framework–UCTR can already achieve promising unsupervised performance on the three datasets. Compared to supervised benchmarks, it reaches 67%, 70%, 87%, 93% of F1 score or label accuracy on the TAT-QA, WiKiSQL, FEVEROUS and SEM-TAB-FACTS, respectively, without using any human-labeled data. Moreover, UCTR outperforms other unsupervised models by large margins. In particular, the F1 score of MQA-QG on TAT-QA is only 27.7, while UCTR achieves 42.4. We suppose the reason is that the data generated by MQA-QG can only cover a small fraction of reasoning types compared to the original dataset, so the trained model cannot handle questions with more complex logical structures. Contrastively, UCTR can take advantage of program templates with various underlying reasoning structures to match the distribution of the original dataset as much as possible.

(2) Our proposed UCTR-ST framework achieves the best performance on all tasks. Compared to the basic version–UCTR, the self-training technique brings significant improvements, which suggests that although synthetic data is abundant, it differs from the distribution of realistic data. After incorporating the distribution information of realistic data by using unlabeled training samples, the model can perform better at testing stage.

(3) Under the few-shot setting, where only 50 labeled instances are available, supervised models perform poorly. In contrast, UCTR-ST gains much better performance with the assistance of a large amount of synthetic data. The results reveal that our method can significantly reduce the labor cost of manual annotation. Additionally, we notice that for FEVEROUS and TAT-QA, models trained on the synthetic

TABLE II
RESULTS ON THE DEVELOPMENT SET OF TAT-QA

Model		Table		Table-Text		Text		Total	
		EM	F1	EM	F1	EM	F1	EM	F1
Supervised	Text-Span only	1.3	1.6	7.7	9.7	47.3	73.5	14.0	20.9
	Table-Cell only	12.0	16.8	20.5	29.2	0.3	1.0	11.9	16.9
	TAPAS [13]	-	-	-	-	-	-	18.9	26.5
	TAGOP [4]	**52.6**	**54.9**	**65.1**	**66.9**	**48.8**	**73.8**	**55.5**	**62.9**
Unsupervised	MQA-QG [30]	9.7	12.4	23.7	30.1	33.2	55.1	19.4	27.7
	UCTR $-w/o$ T2T	28.1	30.0	41.8	47.1	30.6	52.9	32.8	40.5
	UCTR (ours)	30.7	32.4	42.8	47.3	**33.2**	**55.9**	34.9	42.4
	UCTR-ST (ours)	**38.2**	**40.3**	**50.3**	**54.7**	31.1	52.8	**40.2**	**47.6**
Few-Shot	TAGOP [4]	10.4	13.4	11.2	18.6	0.3	0.9	8.3	12.1
	TAGOP+**UCTR-ST**	**42.9**	**45.7**	**58.8**	**62.4**	**44.5**	**72.1**	**48.1**	**56.9**

TABLE III
RESULTS ON FEVEROUS

Model		Dev		Test
		Accuracy	FEVEROUS Score	FEVEROUS Score
Supervised	Sentence-only baseline	81.1	19.0	18.5
	Table-only baseline	81.6	19.1	17.9
	Full baseline [45]	**86.0**	**20.2**	**19.2**
Unsupervised	Random	47.0	14.1	13.2
	MQA-QG [30]	71.1	17.6	16.4
	UCTR (ours)	74.8	18.3	17.0
	UCTR-ST (ours)	**77.7**	**19.7**	**18.3**
Few-Shot	Full baseline [45]	67.3	14.2	13.3
	Full baseline+**UCTR-ST**	**78.2**	**19.7**	**18.4**

TABLE IV
RESULTS ON SEM-TAB-FACTS

Model		3-way micro F1	
		Dev	Test
Supervised	TAPAS [44]	**66.7**	**62.4**
Unsupervised	Random	33.3	33.3
	MQA-QG [30]	53.2	50.4
	TAPAS-Tranfer [44]	59.0	58.7
	UCTR (ours)	62.6	60.3
	UCTR-ST (ours)	**64.2**	**61.2**
Few-Shot	TAPAS [44]	48.6	46.5
	TAPAS+**UCTR-ST**	**64.1**	**61.0**

TABLE V
RESULTS ON WIKISQL

Model		Denotation Accuracy	
		Dev	Test
Supervised	TAPAS [13]	85.1	83.6
	TAPEX [20]	**88.1**	**87.0**
Unsupervised	TAPEX [20]	21.4	21.8
	MQA-QG [30]	57.8	57.2
	UCTR (ours)	62.2	61.6
	UCTR-ST (ours)	**63.5**	**62.7**
Few-Shot	TAPEX [20]	53.8	52.9
	TAPEX+**UCTR-ST**	**63.5**	**62.7**

dataset can gain further improvements by fine-tuning on the 50 high-quality sample s. But for SEM-TAB-FACTS and WiKiSQL, the 50 human-labeled samples don't enhance the model as expected. We suppose it is because the amount of annotated samples is too small to provide additional valuable information on these datasets.

(4) TAPAS-Transfer performs well without fine-tuning on any synthetic samples generated from SEM-TAB-FACTS, which reveals that sufficient training data from the general domain (i.e., the TABFACT dataset) can give a good model initialization for specialized domains. However, TAPAS-Transfer still underperforms our unsupervised framework UCTR-ST. We suppose there are two main reasons. Firstly, the samples of SEM-TAB-FACTS contain lots of scientific terms and numbers. In addition, SEM-TAB-FACTS has one more label–"Unknown" compared to TABFACT, limiting the effectiveness of transfer learning from TABFACT.

TABLE VI
RESULTS OF DATA AUGMENTATION ON TAT-QA, FEVEROUS AND SEM-TAB-FACTS

| Model | | TAT-QA | | SEM-TAB-FACTS | | WiKiSQL | | FEVEROUS |
		Dev	Test	Dev	Test	Dev	Test	Dev
Supervised	Baseline	55.5/62.9	50.1/58.0	66.7	62.4	**88.1**	**87.0**	**86.0**
	Baseline+UCTR	**59.7/67.7**	**56.1/64.3**	**69.8**	**63.9**	87.9	**87.0**	85.9

TABLE VII
ABLATIONS ON THE DEVELOPMENT SET OF TAT-QA

| Setting | Data Source | | | Program Type | | Self Training | Performance | | | |
| | Table | Text | Table↔Text | SQL | Arithmetic | | Table | Table-Text | Text | Total |
							EM / F_1	EM / F_1	EM / F_1	EM / F_1
A1	✓			✓			6.1 / 8.6	17.2 / 21.6	0.8 / 1.5	8.2 / 10.9
A2		✓					1.8 / 2.2	5.3 / 8.2	32.1 / 55.8	10.0 / 16.5
A3	✓	✓		✓			6.3 / 8.4	17.8 / 23.4	31.4 / 54.1	15.7 / 23.6
A4	✓	✓			✓		30.6 / 31.7	35.9 / 38.8	31.8 / 53.0	32.5 / 38.8
A5	✓	✓		✓	✓		28.1 / 30.0	41.8 / 47.1	30.6 / 52.9	32.8 / 40.5
A6	✓	✓	✓	✓	✓		30.7 / 32.4	42.8 / 47.3	33.2 / 55.9	34.9 / 42.4
A7	✓	✓	✓	✓	✓	✓	38.2 / 40.3	50.3 / 54.7	31.1 / 52.8	40.2 / 47.6

D. Data Augmentation.

In this section, we investigate the effectiveness of using our data generation method as a data augmentation technique. Note that we assume that human-labeled training data is available, so we used the basic version–UCTR without using the self-training process. We first fine-tune the model on our synthetic data and then fine-tune it on the high-quality human-labeled data. The performances are shown in Table VI. For TAT-QA, the evaluation metric is the EM and F1 score. For WiKiSQL, the evaluation metric is the denotation accuracy. For FEVEROUS and SEM-TAB-FACTS, the evaluation metric is the label accuracy. Experimental results show that the effectiveness of UCTR varies across different benchmarks. We surprisingly find that UCTR can substantially boost the supervised performance, with a 6.3 absolute gain of F1 score on the test set of TAT-QA and 3.1 gain of label accuracy on the development set of SEM-TAB-FACTS. But similar phenomena are not observed for the FEVEROUS and WiKiSQL.

We suppose the main underlying reason is that UCTR can alleviate the problem of data sparsity. Both TAT-QA and SEM-TAB-FACTS are datasets collected from specialized domains. And the labeled training samples of them are relatively insufficient. Specifically, the number of tables in TAT-QA and SEM-TAB-FACTS are 2,757, 1,085, respectively, compared to over ten thousand tables for FEVEROUS and WiKiSQL. As a result, the data generated by UCTR can make the model get familiar with the tables and provide a good initialization for supervised training.

In summary, our proposed unified framework can generate high-quality human-like data for various tabular reasoning tasks on homogenous or heterogeneous data. The synthetic data can significantly boost the model's performance under an unsupervised or a few-shot setting and even enhance the supervised performance further.

E. Ablation Study.

To evaluate the effectiveness of each component of UCTR-ST, we present the model's performances on the development set of TAT-QA under different ablation settings. The results are depicted in Table VII. "Table↔ Text" under the "Data Source" column means we generate joint table-text reasoning samples using the Table-To-Text operator and Text-To-Table operator. Based on the results, we have the following observations:

From the perspective of data sources, models trained only on tables or texts achieve low performances. In contrast, the model trained on both tabular and textual data obtains the capability of reasoning across modalities and gains considerable improvement. Additionally, the "Table↔ Text" source brings further enhancement, highlighting the ability to reason on a hybrid of tabular and textual data.

From the perspective of program types, arithmetic expressions are more valuable than SQL queries since most samples in TAT-QA require arithmetic operations. The model using all these two types of programs reaches the highest performance.

F. Analysis of Generated Text.

In this section, we present some example sentences generated by the Program-Transformation from different types of programs. The red text spans are important information shared by generated and golden text, while the blue spans are the information mismatched. We can observe that Program-Transformation module can understand the underlying logic and generate appropriate questions or claims. For example, the original arithmetic expression only contains a "subtract" operation followed by a "divide" operation, but the model can identify the meaning of "percentage change" correctly. However, in some cases, the generated text loses some critical information or contains inaccurate information.

TABLE VIII
EXAMPLES EXHIBITING GENERATED TEXT FROM DIFFERENT TYPES OF PROGRAMS. THE RED SPANS ARE THE KEY INFORMATION SHARED BY THE
GENERATED TEXT AND GOLDEN TEXT, WHILE THE SPANS COLORED BY BLUE ARE THE INFORMATION MISMATCHED.

Type	Program	Generated Text	Golden Text
SQL Query	select [department] from table order by [total deputies] desc limit 1	Which department has the most total deputies?	What is the department with the most amount of total deputies?
Logical Form	eq { count { filter_all { all_rows ; Basic Printer Settings Material } } ; 3 }	There are 3 basic printer settings that can be used with a Basic Printer	There are 3 Material used for Basic Printer Settings.
Arithmetic Expression	subtract (the Stockholders' equity of 2019, the Stockholders' equity of 2018) , divide (#0, the Stockholders' equity of 2018)	By what percentage did stockholders' equity decrease from 2018 to 2019?	What was the percentage change in stockholders' equity between 2018 and 2019?

G. Synthetic Data vs. Labeled Data.

In section V-C, we show the few-shot performance of models using only 50 samples. In this section, we conduct a more detailed analysis of the synthetic data and labeled data by changing the number of available labeled samples. Since the synthetic data shows significant effects on TAT-QA according to previous results, we still take the result on the development set of TAT-QA as an example. As shown in Figure 5, the orange line depicts the F1 score of the model first trained on our synthetic data, then further fine-tuned on the available labeled data. In contrast, the blue line shows the performance of the model directly trained on the labeled data.

As the number of samples increases, the model pre-trained on our synthetic data always performs better. In addition, we have several interesting findings: i) The F1 score of the model trained on 23,933 synthetic samples is around 42, comparable to a model trained on 1000 labeled data. (2) When we fine-tune the model trained on 23,933 synthetic samples with additional 1000 human-labeled samples, it can achieve comparable performance to a model trained on 13,217 labeled data. Therefore, we conclude that our unsupervised learning framework provides a good initialization so that the model can gain a considerable improvement using only a small amount of labeled samples. Our framework can be very beneficial in an online learning setting when applying a model to a new domain, where labeled data is limited.

H. Effectiveness of Self-Training

In order to better demonstrate the role of self-training technique in the training process, we show in Figure 6 the performance of the models on four benchmarks as the number of iterations varies. It can be seen that as the model performance improves, the model can generate better pseudo-labels, which in turn can be used to train better models in the next iteration. As the number of iterations increases, the improvement of the model will gradually decrease, and eventually converge. In a nutshell, we verify that our data generation technique can naturally integrate with self-training in a fully unsupervised scenario and achieve good performance on real world data.

VI. RELATED WORK

In this section, we briefly summarize the related works from these two aspects: the development of tabular reasoning models and unsupervised data generation methods.

A. Tabular Reasoning Models.

Many tabular reasoning models tackle the question answering and fact verification tasks in a semantic parsing manner [48], [49], converting a natural language sentence into a program. Zhong et al. [6] translate users' questions to corresponding SQL queries, and Yang et al. [47] generate semantic consistent logical forms with tree structures and execute them to judge the claims. However, the search space for programs is very large, and the model may generate spurious programs which have wrong structures but return the correct answers. Recent works demonstrate that pre-trained language models achieve better reasoning performances on various tasks by pre-training or leveraging auxiliary knowledge [15], [59], [50], [58]. Specifically, for the tabular reasoning task, TAPAS [13] is a BERT-extended model pre-trained on a large corpus of texts and tables from Wikipedia. It answers questions by applying operations on predicted table cells in an end-to-end
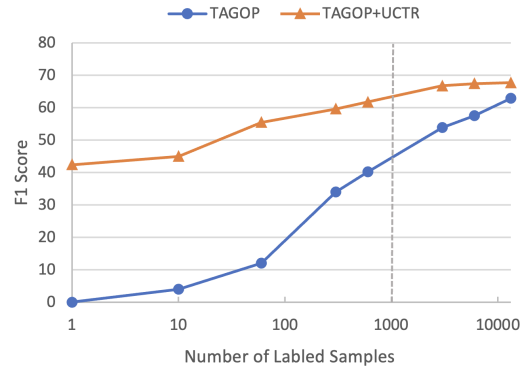

Fig. 5. Effectiveness of the synthetic data. The orange line corresponds to the model first trained on the synthetic data and then fine-tuned on the varied number of labeled samples. The blue line corresponds to the model directly trained on labeled samples.
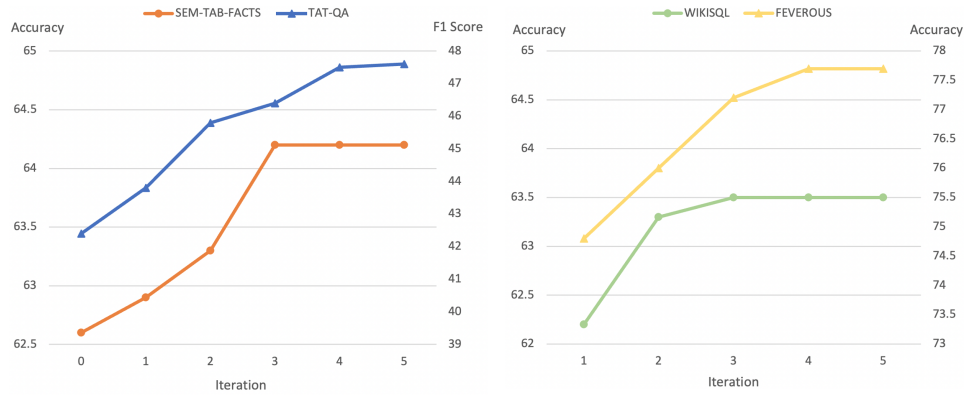
Fig. 6. Effectiveness of the self-training on four benchmarks. We demonstrate how the performance of the model varies with the number of iterations of self-training. It can be observed that the model can self-improve its performance by using only unlabeled data and eventually achieves convergence in about 6 iterations. The final model shows a significant improvement compared to the original model.

way. Neeraja et al. [56] boost the reasoning ability of pre-trained models on the tabular NLI task by introducing external knowledge. And it is a promising direction to explore how to obtain better representations of tables. GraPPa [19] introduces a text-schema linking objective to make the model better understand the grammatical role of table elements. However, the main drawback of these methods is that they require a large amount of training data, limiting their performance when transferring to a new domain.

B. Unsupervised Data Generation and Self-Training.

Unsupervised data generation has been extensively studied on various tasks like question answering and natural language inference, and has shown surprising performances [51], [53], [57]. Recently, methods for synthesizing human-like tabular reasoning samples have also been proposed [52], [54]. Chemmengath et al. [46] sample complex SQL queries and generate natural language questions in a seq2seq manner. Eisenschlos et al. [29] generate factual claims leveraging context-free grammar (CFG) and counterfactual heuristics. Unfortunately, these methods focus on a specific task or scenario. Based on the modules and predefined operators, our approach can convert different types of programs into natural language questions or claims with tabular evidence or a hybrid of tabular and textual evidence.

Self-training has been widely explored in the realm of semi-supervised learning [60], [61], [62]. For example, in the field of table reasoning, Guo et al. [63] proposed MST-SQL, a method that combines self-training and meta-learning to improve the performance of few-shot Text-to-SQL. Most works employ self-training techniques in the few-shot setting, where a small number of labeled samples are available. However, this study effectively combines data generation methods with self-training to achieve good results in an unsupervised scenario.

VII. CONCLUSION AND FUTURE WORKS

We explore the unsupervised complex tabular reasoning task and propose a novel self-training framework UCTR-ST with several optimization techniques. UCTR-ST can synthesize high-quality human-like questions and claims with underlying complex logic without any labeled data and leverage the self-training technique to fully exploit the information of the unlabeled data. Comprehensive experiments for different tasks and domains demonstrate that model achieve surprising performances under unsupervised and few-shot settings, which can significantly ease the burden of human annotation. Moreover, UCTR-ST can boost supervised performances in specialized domains with insufficient data. In future work, we will broaden the reasoning types of programs and explore an auto program-generation method based on the existing data distributions to make the framework more flexible.

REFERENCES

[1] N. X. Wang, D. Mahajan, M. Danilevsky, and S. Rosenthal, "Semeval-2021 task 9: Fact verification and evidence finding for tabular data in scientific documents (sem-tab-facts)," *arXiv preprint arXiv:2105.13995*, 2021.

[2] S. K. Jauhar, P. Turney, and E. Hovy, "Tabmcq: A dataset of general knowledge tables and multiple-choice questions," *arXiv preprint arXiv:1602.03960*, 2016.

[3] Y. Katsis, S. Chemmengath, V. Kumar, S. Bharadwaj, M. Canim, M. Glass, A. Gliozzo, F. Pan, J. Sen, K. Sankaranarayanan *et al.*, "Ait-qa: Question answering dataset over complex tables in the airline industry," *arXiv preprint arXiv:2106.12944*, 2021.

[4] F. Zhu, W. Lei, Y. Huang, C. Wang, S. Zhang, J. Lv, F. Feng, and T.-S. Chua, "Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance," *arXiv preprint arXiv:2105.07624*, 2021.

[5] W. Chen, H. Wang, J. Chen, Y. Zhang, H. Wang, S. Li, X. Zhou, and W. Y. Wang, "Tabfact: A large-scale dataset for table-based fact verification," *arXiv preprint arXiv:1909.02164*, 2019.

[6] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[7] P. Pasupat and P. Liang, "Compositional semantic parsing on semi-structured tables," *arXiv preprint arXiv:1508.00305*, 2015.

[8] V. Gupta, M. Mehta, P. Nokhiz, and V. Srikumar, "Infotabs: Inference on tables as semi-structured data," *arXiv preprint arXiv:2005.06117*, 2020.

[9] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," *arXiv preprint arXiv:1508.05326*, 2015.

[10] I. Dagan, O. Glickman, and B. Magnini, "The pascal recognising textual entailment challenge," in *Machine learning challenges workshop*. Springer, 2005, pp. 177–190.

[11] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[12] Z. Wang, H. Dong, R. Jia, J. Li, Z. Fu, S. Han, and D. Zhang, "Tuta: tree-based transformers for generally structured table pre-training," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1780–1790.

[13] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. M. Eisenschlos, "Tapas: Weakly supervised table parsing via pre-training," *arXiv preprint arXiv:2004.02349*, 2020.

[14] A. Nassar, N. Livathinos, M. Lysak, and P. Staar, "Tableformer: Table structure understanding with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4614–4623.

[15] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel, "Tabert: Pretraining for joint understanding of textual and tabular data," *arXiv preprint arXiv:2005.08314*, 2020.

[16] S. Ö. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, 2021, pp. 6679–6687.

[17] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu, "Turl: Table understanding through representation learning," *ACM SIGMOD Record*, vol. 51, no. 1, pp. 33–40, 2022.

[18] J. M. Eisenschlos, M. Gor, T. Müller, and W. W. Cohen, "Mate: Multi-view attention for table transformer efficiency," *arXiv preprint arXiv:2109.04312*, 2021.

[19] T. Yu, C.-S. Wu, X. V. Lin, B. Wang, Y. C. Tan, X. Yang, D. Radev, R. Socher, and C. Xiong, "Grappa: grammar-augmented pre-training for table semantic parsing," *arXiv preprint arXiv:2009.13845*, 2020.

[20] Q. Liu, B. Chen, J. Guo, Z. Lin, and J.-g. Lou, "Tapex: Table pre-training via learning a neural sql executor," *arXiv preprint arXiv:2107.07653*, 2021.

[21] T. Xie, C. H. Wu, P. Shi, R. Zhong, T. Scholak, M. Yasunaga, C.-S. Wu, M. Zhong, P. Yin, S. I. Wang *et al.*, "Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models," *arXiv preprint arXiv:2201.05966*, 2022.

[22] D. Choi, M. C. Shin, E. Kim, and D. R. Shin, "Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases," *Computational Linguistics*, vol. 47, no. 2, pp. 309–332, 2021.

[23] B. Wang, I. Titov, and M. Lapata, "Learning semantic parsers from denotations with latent structured alignments and abstract programs," *arXiv preprint arXiv:1909.04165*, 2019.

[24] W. Hwang, J. Yim, S. Park, and M. Seo, "A comprehensive exploration on wikisql with table-aware word contextualization," *arXiv preprint arXiv:1902.01069*, 2019.

[25] Y. Zhao, Y. Li, C. Li, and R. Zhang, "Multihiertt: Numerical reasoning over multi hierarchical tabular and textual data," *arXiv preprint arXiv:2206.01347*, 2022.

[26] H. Iida, D. Thai, V. Manjunatha, and M. Iyyer, "Tabbie: Pretrained representations of tabular data," *arXiv preprint arXiv:2105.02584*, 2021.

[27] H. Gong, Y. Sun, X. Feng, B. Qin, W. Bi, X. Liu, and T. Liu, "Tablegpt: Few-shot table-to-text generation with table structure reconstruction and content matching," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 1978–1988.

[28] Z. Chen, W. Chen, C. Smiley, S. Shah, I. Borova, D. Langdon, R. Moussa, M. Beane, T.-H. Huang, B. Routledge *et al.*, "Finqa: A dataset of numerical reasoning over financial data," *arXiv preprint arXiv:2109.00122*, 2021.

[29] J. M. Eisenschlos, S. Krichene, and T. Müller, "Understanding tables with intermediate pre-training," *arXiv preprint arXiv:2010.00571*, 2020.

[30] L. Pan, W. Chen, W. Xiong, M.-Y. Kan, and W. Y. Wang, "Unsupervised multi-hop question answering by question generation," *arXiv preprint arXiv:2010.12623*, 2020.

[31] H. Dong, Z. Cheng, X. He, M. Zhou, A. Zhou, F. Zhou, A. Liu, S. Han, and D. Zhang, "Table pretraining: A survey on model architectures, pretraining objectives, and downstream tasks," *arXiv preprint arXiv:2201.09745*, 2022.

[32] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "Fever: a large-scale dataset for fact extraction and verification," *arXiv preprint arXiv:1803.05355*, 2018.

[33] A. Judea, H. Schütze, and S. Brügmann, "Unsupervised training set generation for automatic acquisition of technical terminology in patents," in *Proceedings of COLING 2014, the 25th international conference on computational linguistics: Technical Papers*, 2014, pp. 290–300.

[34] X. Pi, Q. Liu, B. Chen, M. Ziyadi, Z. Lin, Y. Gao, Q. Fu, J.-G. Lou, and W. Chen, "Reasoning like program executors," *arXiv preprint arXiv:2201.11473*, 2022.

[35] Z. Chen, W. Chen, H. Zha, X. Zhou, Y. Zhang, S. Sundaresan, and W. Y. Wang, "Logic2text: High-fidelity natural language generation from logical forms," *arXiv preprint arXiv:2004.14579*, 2020.

[36] M. Kale and A. Rastogi, "Text-to-text pre-training for data-to-text tasks," *arXiv preprint arXiv:2005.10433*, 2020.

[37] Y. Su, D. Vandyke, S. Wang, Y. Fang, and N. Collier, "Plan-then-generate: Controlled data-to-text generation via planning," *arXiv preprint arXiv:2108.13740*, 2021.

[38] X. Wu, J. Zhang, and H. Li, "Text-to-table: A new way of information extraction," *arXiv preprint arXiv:2109.02707*, 2021.

[39] T. Shi, C. Zhao, J. Boyd-Graber, H. Daumé III, and L. Lee, "On the potential of lexico-logical alignments for semantic parsing to sql queries," *arXiv preprint arXiv:2010.11246*, 2020.

[40] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[41] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[42] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[43] P. Li, W. Li, Z. He, X. Wang, Y. Cao, J. Zhou, and W. Xu, "Dataset and neural recurrent sequence labeling model for open-domain factoid question answering," *arXiv preprint arXiv:1607.06275*, 2016.

[44] D. Gautam, K. Gupta, and M. Shrivastava, "Volta at semeval-2021 task 9: Statement verification and evidence finding with tables using tapas and transfer learning," *arXiv preprint arXiv:2106.00248*, 2021.

[45] R. Aly, Z. Guo, M. Schlichtkrull, J. Thorne, A. Vlachos, C. Christodoulopoulos, O. Cocarascu, and A. Mittal, "Feverous: Fact extraction and verification over unstructured and structured information," *arXiv preprint arXiv:2106.05707*, 2021.

[46] S. A. Chemmengath, V. Kumar, S. Bharadwaj, J. Sen, M. Canim, S. Chakrabarti, A. Gliozzo, and K. Sankaranarayanan, "Topic transferable table question answering," *arXiv preprint arXiv:2109.07377*, 2021.

[47] X. Yang, F. Nie, Y. Feng, Q. Liu, Z. Chen, and X. Zhu, "Program enhanced fact verification with verbalization and graph attention network," *arXiv preprint arXiv:2010.03084*, 2020.

[48] K. Guu, P. Pasupat, E. Z. Liu, and P. Liang, "From language to programs: Bridging reinforcement learning and maximum marginal likelihood," *arXiv preprint arXiv:1704.07926*, 2017.

[49] C. Liang, J. Berant, Q. Le, K. D. Forbus, and N. Lao, "Neural symbolic machines: Learning semantic parsers on freebase with weak supervision," *arXiv preprint arXiv:1611.00020*, 2016.

[50] M. Glass, M. Canim, A. Gliozzo, S. Chemmengath, V. Kumar, R. Chakravarti, A. Sil, F. Pan, S. Bharadwaj, and N. R. Fauceglia, "Capturing row and column semantics in transformer based question answering over tables," *arXiv preprint arXiv:2104.08303*, 2021.

[51] N. Varshney, P. Banerjee, T. Gokhale, and C. Baral, "Unsupervised natural language inference using phl triplet generation," *arXiv preprint arXiv:2110.08438*, 2021.

[52] D. Guo, Y. Sun, D. Tang, N. Duan, J. Yin, H. Chi, J. Cao, P. Chen, and M. Zhou, "Question generation from sql queries improves neural semantic parsing," *arXiv preprint arXiv:1808.06304*, 2018.

[53] S. Shakeri, C. N. d. Santos, H. Zhu, P. Ng, F. Nan, Z. Wang, R. Nallapati, and B. Xiang, "End-to-end synthetic data generation for domain adaptation of question answering systems," *arXiv preprint arXiv:2010.06028*, 2020.

[54] I. V. Serban, A. García-Durán, C. Gulcehre, S. Ahn, S. Chandar, A. Courville, and Y. Bengio, "Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus," *arXiv preprint arXiv:1603.06807*, 2016.

[55] C. Malon, "Team papelo at feverous: Multi-hop evidence pursuit," in *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, 2021, pp. 40–49.

[56] J. Neeraja, V. Gupta, and V. Srikumar, "Incorporating external knowledge to enhance tabular reasoning," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 2799–2809.

[57] X. Li, Z. Li, Z. Zhang, N. Liu, H. Yuan, W. Zhang, Z. Liu, and J. Wang, "Effective few-shot named entity linking by meta-learning," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 2022, pp. 178–191.

[58] Z. Duan, X. Li, Z. Li, Z. Wang, and J. Wang, "Not just plain text! fuel document-level relation extraction with explicit syntax refinement and subsentence modeling," *arXiv preprint arXiv:2211.05343*, 2022.

[59] X. Li, Z. Li, Z. Duan, J. Xu, N. Liu, and J. Wang, "Jointly modeling fact triples and text information for knowledge base completion," in *2021 IEEE International Conference on Big Knowledge (ICBK)*. IEEE, 2021, pp. 214–221.

[60] M.-R. Amini and P. Gallinari, "Semi-supervised logistic regression," in *ECAI*, vol. 2, no. 4, 2002, p. 11.

[61] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," *Advances in neural information processing systems*, vol. 17, 2004.

[62] D.-H. Lee *et al.*, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, vol. 3, no. 2, 2013, p. 896.

[63] X. Guo, Y. Chen, G. Qi, T. Wu, and H. Xu, "Improving few-shot text-to-sql with meta self-training via column specificity," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI*, 2022, pp. 4150–4156.