UniRPG: Unified Discrete Reasoning over Table and Text as Program Generation

Yongwei Zhou[1], Junwei Bao[2]∗, Chaoqun Duan[2], Youzheng Wu[2],
Xiaodong He[2], Tiejun Zhao[1]
[1]Harbin Institute of Technology   [2]JD AI Research
ywzhou@hit-mtlab.net   baojunwei001@gmail.com   tjzhao@hit.edu.cn

Abstract

Question answering requiring discrete reasoning, e.g., arithmetic computing, comparison, and counting, over knowledge is a challenging task. In this paper, we propose **UniRPG**, a semantic-parsing-based approach advanced in interpretability and scalability, to perform **Uni**fied **d**iscrete **R**easoning over heterogeneous knowledge resources, i.e., table and text, as **P**rogram **G**eneration. Concretely, UniRPG consists of a neural programmer and a symbolic program executor, where a program is the composition of a set of pre-defined general atomic and higher-order operations and arguments extracted from table and text. First, the programmer parses a question into a program by generating operations and copying arguments, and then, the executor derives answers from table and text based on the program. To alleviate the costly program annotation issue, we design a distant supervision approach for programmer learning, where pseudo programs are automatically constructed without annotated derivations. Extensive experiments on the TAT-QA dataset show that UniRPG achieves tremendous improvements and enhances interpretability and scalability compared with previous state-of-the-art methods, even without derivation annotation. Moreover, it achieves promising performance on the textual dataset DROP without derivation annotation. [1]

1 Introduction

Question answering requiring discrete reasoning, e.g., arithmetic computing, sorting, comparison, and counting, over knowledge is a challenging task and has attracted wide attention. Advanced approaches to this task are roughly divided into mixture-of-experts (MoE)-based and semantic parsing (SP)-based methods. The former divides questions into several limited categories according to

---

∗Work was done during the first author's internship at JD AI mentored by Junwei Bao: baojunwei001@gmail.com.

[1]The code is released at https://github.com/JD-AI-Research-NLP/UniRPG.

Figure 1: The upper block shows a training instance from the TAT-QA dataset. It consists of a table $\mathcal{T}$, question $\mathcal{Q}$, paragraphs $\mathcal{P}$, answer $\mathcal{A}$ and derivation $\mathcal{D}$. The bottom block displays the constructed programs $\mathcal{G}$ under w/ and w/o derivation settings. The gold program is in red, and the others are in green.

their required reasoning type and designs multiple specialized predictors to deal with each category (Dua et al., 2019; Ran et al., 2019; Chen et al., 2020a; Zhu et al., 2021; Zhou et al., 2022a). The latter translates questions into programs such as SPARQL queries (Bao et al., 2014, 2016; Chen et al., 2020d; Gan et al., 2021; Cao et al., 2021; Hui et al., 2022), to derive the answers. Compared with MoE-based methods, SP-based methods have advantages in interpretability and scalability, and we mainly explore discrete reasoning along this technical route in this paper.

As to works based on SP-based methods, they always research over homogeneous knowledge resource, i.e., either unstructured text (Dua et al., 2019; Ran et al., 2019; Chen et al., 2020a; Zhou et al., 2022a) or structured tables and knowledge graphs (Pasupat and Liang, 2015; Herzig et al., 2020; Liu et al., 2022). Respectively, they design

a specialized semantic parser for each knowledge category according to the corresponding data characteristic. However, in practical scenarios (e.g., financial reports and sports news analysis), it usually requires a model to comprehensively understand heterogeneous knowledge resources (e.g., tables and text) and perform reasoning over them. Therefore, to extend SP-based methods to more general and realistic scenarios, it needs a unified framework for discrete reasoning over heterogeneous knowledge resources. Meanwhile, to mitigate the program annotation cost, it also needs an automatic approach to produce pseudo programs.

To address the above challenges, we propose **UniRPG**, a SP-based approach, to perform **Un**ified discrete **R**easoning over table and text as **P**rogram **G**eneration. Specifically, UniRPG consists of a neural programmer and a deterministic symbolic program executor, where a program is the composition of a set of pre-defined general atomic and higher-order operations and arguments extracted from table and text. The programmer leverages a structure-aware knowledge reader to encode heterogeneous knowledge, and a program generator to decode programs through generating operations and copying arguments from table and text. During inference, decoding constraints are leveraged to ensure program legality. Then, the executor derives answers from table and text based on the program. To alleviate the costly program annotation issue, we design a distant supervision approach for programmer learning, where pseudo programs are automatically constructed without annotated derivations. Furthermore, a re-weight strategy is introduced to measure the importance of pseudo programs for noise reduction in training.

To verify the effectiveness and generality of UniRPG, we conduct comprehensive experiments on the TAT-QA (Zhu et al., 2021) and DROP (Dua et al., 2019) datasets under w/ and w/o derivation settings. Experiment results show that UniRPG achieves tremendous improvements and enhances interpretability and scalability compared with previous state-of-the-art (SOTA) methods on the TAT-QA dataset. Specifically, it outperforms the previous SOTA model by 17.0 EM and 18.0 F1 on the test set under full supervision. Besides, our weakly supervised model based on automatically constructed pseudo-programs only lags behind fully supervised model 1.0 EM and 1.4 F1 points. Moreover, it also achieves promising performance

on the textual dataset DROP in weak supervision. In conclusion, the contributions of this work are summarized as follows: (1) We propose an effective program-generation-based framework for discrete reasoning over table and text; (2) To alleviate costly program annotation, we design a distant supervision approach to construct pseudo programs automatically; (3) We conduct comprehensive experiments to verify the effectiveness and scalability of UniRPG on the TAT-QA and DROP datasets.

2 Methodology

2.1 Task Definition

The task of discrete reasoning over hybrid data aims to predict the answer from the given input (e.g., tables and text). Formally, given a question $\mathcal{Q}$, a table $\mathcal{T}$, and a set of paragraphs $\mathcal{P}$, the target is to derive the answer $\mathcal{A}^*$ from $\mathcal{T}$ and $\mathcal{P}$.

$$\mathcal{A}^* = \arg\max P(\mathcal{A}|\mathcal{Q}, \mathcal{T}, \mathcal{P}; \theta). \quad (1)$$

Specifically, the table $\mathcal{T}$ consists of $m \times n$ cells $\{c_{ij}\}$, where m and n are the number of rows and columns. The paragraph set $\mathcal{P} = \{p_1, p_2, ..., p_k\}$ contains k paragraphs which are relevant to the table. Notably, the answer $\mathcal{A}^*$ in this task is either a span extracted from the table and the paragraphs or a number computed based on them. Therefore, it requires the model to perform discrete reasoning such as arithmetic operations, counting, comparison, sorting, and their compositions.

2.2 Framework Overview

As mentioned above, the method of this task needs to be capable of performing discrete reasoning over heterogeneous data. To this end, we first define a set of operations as reasoning units, which consists of atomic and high-order operations. Based on these operations, we propose to perform unified discrete reasoning over table and text as program generation (UniRPG). Concretely, UniRPG consists of a neural programmer and a symbolic program executor, and it can be formulated as:

$$P(\mathcal{A}|\mathcal{Q}, \mathcal{T}, \mathcal{P}; \theta)$$
$$= \sum_{\mathcal{G}} \mathbb{I}(f(\mathcal{G}) = \mathcal{A}) P(\mathcal{G}|\mathcal{Q}, \mathcal{T}, \mathcal{P}; \theta), \quad (2)$$

where $P(\mathcal{G}|\mathcal{Q}, \mathcal{T}, \mathcal{P}; \theta)$ denotes the *neural programmer* and $f(\cdot)$ is the symbolic *program executor*. $\mathbb{I}(f(\mathcal{G}) = \mathcal{A})$ is an indicator function with value 1, if the answer $\mathcal{A}$ can be derived based on the program $\mathcal{G}$, and 0 otherwise.

Operations	Arguments	Descriptions
Atomic Operations		
SPAN	(s, e)	
CELL	(s, e)	Extract a piece of text or a number from table and
VALUE	(s, e)	paragraphs based on the index of start(s) and end(e) tokens.
CELL_VALUE (CV)	(s, e)	
Higher-order Operations		
KV	$\langle$CELL, CV$\rangle$, $\langle$SPAN, VALUE$\rangle$	Build a key-value pair with a text piece and a number.
COUNT	Atomic Operations	Return the number of extracted text pieces/numbers.
MULTI_SPANS	Atomic Operations	Return all the extracted text pieces/numbers.
ARGMAX/ARGMIN	KV	Return the key with maximum/minimum value from all the key-value pair candidates.
SUM/DIFF/TIMES/DIV	VALUE, CV, Constants	Compute the sum/difference/product/quotient between two numbers.
AVG	VALUE, CV, Constants	Compute the average value of argument numbers.
CHANGE_R	VALUE, CV, Constants	Compute the rate of change between two numbers.
Constants		
0 / 1 / 100	-	Several constants are commonly involved in math problems.

Table 1: All the defined operations, their arguments and descriptions.

2.3 Operations

As shown in Table 1, we define 15 operations, which consist of 4 atomic operations and 11 high-order operations. The atomic operations indicate extracting a piece of text or a number from the table or paragraphs. Specifically, SPAN and VALUE are used to extract content from paragraphs while CELL and CELL_VALUE(CV) are used to extract content from the table. The atomic operations are the foundation for unified discrete reasoning over table and text. As to high-order operations, they indicate how to process the arguments extracted based on atomic operations. For example, KV, ARGMAX, and ARGMIN are tailored for questions requiring comparison or sorting. COUNT and MULTI_SPANS are used to solve counting and multi-spans extraction problems, respectively. We also introduce some arithmetic operations to deal with math questions, including SUM, DIFF, TIMES, DIV, AVG and CHANGE_R. Therefore, we formulate the priority of atomic operations higher than high-order operations. In addition, we also propose some constants that are commonly used in math problems.

2.4 Neural Programmer

In this work, we adopt the BART (Lewis et al., 2020) to implement the Neural Programmer. To adapt it to this task, we have improved its encoder and decoder respectively.

2.4.1 Structure-Aware Knowledge Reader

The original encoder of BART only takes textual data as input. However, in this task, besides paragraphs, the input contains tables as well. Following previous works (Liu et al., 2022; Zhou et al., 2022b; Zhang et al., 2020), we first flatten the table into a sequence and concatenate it with the paragraphs together, and then feed the processed paragraphs and table into the encoder. Formally, we rank paragraphs based on the similarity between each one with the question. As to the table, we flatten it by rows and then construct the input as follows:

$$\mathcal{S} = [\langle \mathrm{s} \rangle; \mathcal{Q}; \langle /\mathrm{s} \rangle; \mathcal{T}; \langle /\mathrm{s} \rangle; \mathcal{P}; \langle /\mathrm{s} \rangle]. \quad (3)$$

Then, we feed it into the encoder to learn the hybrid knowledge representation $\mathbf{H}^k = \{\mathbf{h}_i^k\}_{i=1}^L$, where L is the length of the input sequence.

Unlike the text that only contains the linear structure, tables have a more complex 2D structure. Therefore, simple concatenation can not completely capture the structural information in the table. Inspired by Zhang et al. (2020), we inject table structure information into self-attention layers with a structure-aware attention mask. Specifically, in the lower layers, each cell only takes information from other cells in the same row, question, and paragraphs into account. In the upper layers, the information of cells in the same column is also integrated into the representation of the target cell. This way, the encoder can learn the lexical features in the lower layers and capture complex cross-row reasoning in the upper layers. Finally, we denote this encoder as a structure-aware knowledge reader.

2.4.2 Programmer Generator

In this work, the program generator is responsible for generating programs that are used to perform discrete reasoning over table and text. Specifically,
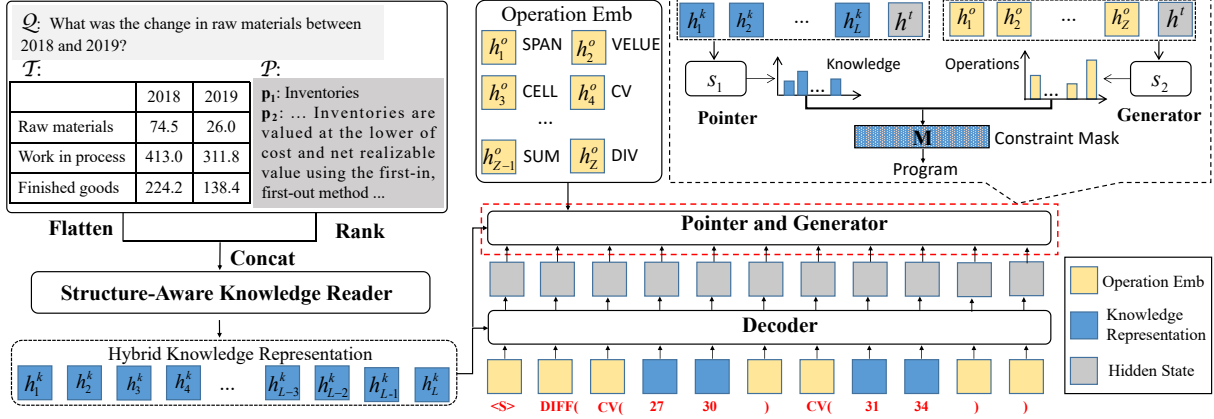
Figure 2: The architecture of neural programmer. It consists of a structure-aware knowledge reader and a program generator. $\mathbf{M}$ is a decoding constraint mask to ensure program legality.

an executable program contains several operations. For the atomic operations, they need to take the start and end indexes of the content in the text and the table as input. For the high-order operations, they take the output of the atomic operations as input. Therefore, the generator should decode the program based on operations, text, and table. However, the original decoder of the BART (Lewis et al., 2020) is used to generate sentences. To bridge the gap, we substitute the prediction vocabulary with the operation set and integrate the copy mechanism (Gu et al., 2016; See et al., 2017) into the BART to retrieve reasoning-related numbers or text pieces from table and text as operation arguments when decoding programs.

Formally, we extra define a trainable embedding matrix $\{\mathbf{h}_j^o\}_{j=1}^Z$ to indicate the operations and the j-th row corresponds to the j-th operation $o_j \in \mathcal{O}$. Based on the decoder of the BART, we can compute two sets of scores at each step as follows:

$$s_i^{k,t} = s_1(\mathbf{h}^t, \mathbf{h}_i^k), i \in [1, 2, 3, ..., L], \quad (4)$$

$$s_j^{o,t} = s_2(\mathbf{h}^t, \mathbf{h}_j^o), j \in [1, 2, ..., Z], \quad (5)$$

where $s_i^{k,t}$ and $s_j^{o,t}$ denote the scores of the i-th token of the input and j-th operation. $\mathbf{h}^t$ is the hidden state of the decoder at the t-th step. $\mathbf{h}_i^k$ is the contextual representation of the i-th token in the input. $s_1(\cdot)$ and $s_2(\cdot)$ are both cosine similarity functions. $s_1(\cdot)$ is a pointer to locate the copied numbers and text pieces from the input sequence $\mathcal{S}$. $s_2(\cdot)$ acts as a generator to select an operation from the operation set $\mathcal{O}$. After obtaining these two scores, we normalize them with a softmax function.

2.5 Program Executor

In this work, the program is executable, and we can directly apply it to perform discrete reasoning over table and text. Therefore, we implement a symbolic program executor rather than a specific neural module. Given a program, we can execute it based on the meaning of each operation in it. For example, given a program $\texttt{SUM}(\texttt{CV}(\texttt{s}_1, \texttt{e}_1), \texttt{CV}(\texttt{s}_2, \texttt{e}_2))$, since the priority of atomic operations is higher than high-order operations, $\texttt{CV}(\texttt{s}_1, \texttt{e}_1)$ and $\texttt{CV}(\texttt{s}_2, \texttt{e}_2)$ are first executed to extract value from the table. And then, $\texttt{SUM}(\cdot)$ is executed to return the summation of extracted value from the table as the output.

3 Training and Inference

3.1 Training

Training Instance Construction: w/ Derivation
Each training instance $\mathcal{I} = \{\mathcal{Q}, \mathcal{T}, \mathcal{P}, \mathcal{A}, \mathcal{D}\}$ consists of a question $\mathcal{Q}$, a table $\mathcal{T}$, associated paragraphs $\mathcal{P}$, an answer $\mathcal{A}$ and a derivation $\mathcal{D}$. The derivation $\mathcal{D}$ indicates the answer position in table and text and the reasoning solution to derive the answer. We can deterministically construct a program based on such a training instance $\mathcal{I}$ that contains annotated derivation $\mathcal{D}$. As the example shown in Figure 1, we determine the unique program $\mathcal{G} = \texttt{DIFF}(\texttt{CV}(131, 134), \texttt{CV}(135, 138))$ based on the reasoning solution $0.06 = 0.53 - 0.47$ and the positions of these two numbers.

Training Instance Construction: w/o Derivation
In addition, to alleviate the costly derivation annotation, we design a distant supervision method to construct pseudo programs automatically without derivation. We design a template library $\mathcal{B}$ shown

7497

in Table 9 in the Appendix A.3, which consists of five categories, namely text/number extraction, multi-spans extraction, counting, comparison/sorting, and arithmetic computing. We search all the possible programs for each training instance based on $\{\mathcal{Q}, \mathcal{T}, \mathcal{P}, \mathcal{A}, \mathcal{B}\}$. For text/number extraction, we find all the occurrences of answers from table and text. Concerning multi-spans extraction, we find all the occurrences of each span and then combine them. For counting questions, building a fine-grained counting program fails due to a lack of the positions of answer clues. To address the issue, we convert a question with a multi-spans answer into a counting question by replacing its interrogatives (e.g. "What", "Which", "Who") with "How many", and leverage its answer items to build the counting programs. For questions requiring comparison/sorting, we first find the answer position in the table and construct key-value pairs with cells in the same row/column and cells in other rows/columns that contain values. If performing the ARGMAX/ARGMIN operation on the list of key-value pairs yields the answer, we then append it to the program candidates. Additionally, we define several math templates that conduct specific operations on numbers in table and text for math problems. Afterward, for each training instance in TAT-QA, more programs (eight on average) that can derive correct answers are found without using annotated derivation $\mathcal{D}$. For instance, another program DIFF(CV(131, 134), CV(135, 138)) is found under w/o derivation setting shown in Figure 1. Moreover, based on the templates, we find at least one pseudo program for 89% training instances in the TAT-QA dataset (Zhu et al., 2021).

Training Objective We consider all the programs as supervision constructed under the w/ derivation setting. For w/o derivation setting, we introduce a re-weight strategy to measure the importance of pseudo programs for noise reduction in training. Formally, the program generation loss $\mathcal{L}_g$ is defined as follows:

$$\mathcal{L}_g = -\sum_{\mathcal{G} \in \Omega} \alpha_{\mathcal{G}} \log p(\mathcal{G}|\mathcal{Q}, \mathcal{T}, \mathcal{P}; \theta). \quad (6)$$

Precisely, we calculate a weight $\alpha_{\mathcal{G}}$ for each pseudo program $\mathcal{G} \in \Omega$ as the reciprocal of the number of pseudo programs with the same operations. Based on the weight distribution, losses of pseudo programs are weighted summed to treat each type of program fairly. In addition, TAT-QA has another

Dataset	Derivation	Train	Dev	Test	Total
TAT-QA					
#question	Yes	13215	1668	1669	16552
#hybrid contexts		2201	278	278	2757
DROP					
#question	No	77409	9536	9622	96567
#passage		5565	582	588	6735

Table 2: The statics of the two datasets. # means the number of each item.

task to predict the scale of answer and it may be *None*, *Thousand*, *Million*, *Billion* and *Percent*. A correct final answer requires that both the predicted answer text and scale are correct. We propose a 5-categories classifier for scale prediction and joint train it with the neural programmer. Denote the cross-entropy loss for scale prediction as $\mathcal{L}_s$ and then the total loss can be calculated with a weight λ as $\mathcal{L} = \mathcal{L}_g + \lambda \mathcal{L}_s$.

3.2 Inference with Constraint Decoding

To ensure the legality of the program, we propose the following three categories of decoding constraints and utilize a decoding mask to filter the illegal program candidates at each step. (1) *Index constraints*. For example, with (s, e) as arguments to an atomic operation, the end index e must be equal to or larger than the start index s. (2) *Type constraints*. For example, the operation CV should return a number from the table rather than paragraphs. (3) *Composition constraints*. Each operation must take the correct type of operations as arguments. More decoding constraint details are shown in Appendix A.2.

4 Experiment

4.1 Dataset and Evaluation Metrics

The experiments are conducted on two QA datasets, including TAT-QA (Zhu et al., 2021) and DROP (Dua et al., 2019). TAT-QA is a large-scale QA dataset in finance created to facilitate discrete reasoning over table and text. DROP is a large-scale QA dataset requiring discrete reasoning over text. It is crowd-sourced based on passages from Wikipedia. The TAT-QA dataset especially provides the derivation for each training instance, but the DROP dataset does not. The statics information of the two datasets is summarized in Table 2. We employ Exact Match (EM) and F1 as the evaluation metrics.

4.2 Implementation Details

Note that the implementation details are described in the Appendix A.1.

4.3 Baselines

Baselines for TAT-QA (1) *Textual QA Models:* BERT-RC (Devlin et al., 2019) is a simple extractive reading comprehension (RC) model that predicts a text span as the answer. NumNet+v2 (Ran et al., 2019) leverages a graph neural network to perform reasoning over numbers in tables and paragraphs. When applying RC models to TAT-QA, the table is flattened into a sequence by row and concatenated with questions and paragraphs. (2) *Tabular QA Models:* The baseline TaPas for WTQ employs TaPas (Herzig et al., 2020) as the knowledge encoder. TaPas contains prior knowledge by pre-training over large-scale tabular data and injects table structure information in the embedding layer. (3) *Hybrid QA Models:* HyBrider (Chen et al., 2020c) first links the question with a table cell and feeds paragraphs and the selected cell into the RC module to infer the answer. TAGOP (Zhu et al., 2021) is an MoE-based model that defines multiple operations as predictors to support discrete reasoning. It first detects the evidence from tables and paragraphs and selects an answer predictor to infer the answer based on the evidence.

Baselines for DROP NAQANet (Dua et al., 2019) designs three specific modules to handle different types of questions, including span extraction, counting, and arithmetic calculation. MTMSN (Hu et al., 2019) introduces an extra answer predictor to address negation questions and a re-ranking mechanism to rank arithmetic expression candidates considering their context information. NumNet (Ran et al., 2019) builds a bi-directional fully-connected graph to model the magnitude of numbers and employs a graph neural network to reason over it. GenBERT (Geva et al., 2020) injects numerical reasoning capability into BERT by pre-training with large-scale synthetic numerical data.

4.4 Main Results

Result on TAT-QA Table 3 displays the experiment results of baselines and UniRPG on the TAT-QA dataset. All the baseline results come from previous work (Zhu et al., 2021). We implement UniRPG based on BART (Lewis et al., 2020) and train it under the w/ and w/o derivation settings, respectively. As shown in Table 3, UniRPG(BART-base) and UniRPG(BART-large) respectively exceed the previous SOTA method TAGOP by 14.3 EM/15.6 F1 and 17.0 EM/18.0 F1 on the test set under full supervision (w/ derivation).

Method	Dev		Test	
	EM	F1	EM	F1
Baselines (w/ derivation)				
BERT-RC (Devlin et al., 2019)	9.5	17.9	9.1	18.7
NumNet+v2 (Ran et al., 2019)	38.1	48.3	37.0	46.9
TaPas for WTQ (Herzig et al., 2020)	18.9	26.5	16.6	22.8
HyBrider (Chen et al., 2020c)	6.6	8.3	6.3	7.5
TAGOP (Zhu et al., 2021)	55.2	62.7	50.1	58.0
w/o derivation				
UniRPG(BART-base)	66.1	73.4	61.5	70.8
UniRPG(BART-large)	**69.1**	**76.4**	**66.1**	**74.6**
w/ derivation				
UniRPG(BART-base)	67.5	74.8	64.4	73.6
UniRPG(BART-large)	**70.2**	**77.9**	**67.1**	**76.0**
Human			84.1	90.8

Table 3: Results of baselines and our models on the TAT-QA dataset.

Method	Dev		Test	
	EM	F1	EM	F1
NAQANet (Dua et al., 2019)	46.20	49.24	44.07	47.01
NumNet (Ran et al., 2019)	64.92	68.31	64.56	67.97
GenBERT (Geva et al., 2020)	68.80	72.30	68.60	72.35
MTMSN (Hu et al., 2019)	76.68	80.54	75.88	79.99
UniRPG	**78.02**	**80.93**	**77.08**	**80.42**

Table 4: Results on the DROP dataset.

In addition, our framework also works well under weak supervision (w/o derivation), exceeding all the baselines by a remarkable margin. Moreover, our weakly-supervised model based on automatically constructed pseudo programs only lags behind fully-supervised models 1.0 EM and 1.4 F1 on the test set. We think the promising improvement comes from the following aspects: (1) UniRPG can solve more complex and more types of questions, such as compositional reasoning and abstract answers (answers do not appear in context). (2) As visualized in Figure 3, UniRPG captures the relationship between programs and hybrid knowledge.

Moreover, Table 5 displays the detailed experiment results of baseline (TAGOP) and our UniRPG(BART-large) under w/ derivation setting on the test set w.r.t answer types and sources. Our method significantly outperforms TAGOP, whether the answers come from table, text, and collaboration of table and text. Furthermore, UniRPG has advantages in the following three types of answers, including Span, Spans, and Arithmetic answers, yet disadvantages in Counting answers. We speculate the phenomenon stems from the lack of training instances with Counting answers, which only account for 2%.

	Table	Text	Table-Text	Total
Span	57.8 ǀ 67.6	70.6 ǀ 79.5	71.7 ǀ 85.4	67.9 ǀ **78.0**
Spans	77.0 ǀ 84.1	59.1 ǀ 60.2	76.9 ǀ 78.9	75.1 ǀ **78.5**
Counting	63.6 ǀ 18.1	- ǀ -	62.1 ǀ 31.0	**62.5** ǀ 27.0
Arithmetic	41.1 ǀ 75.8	27.3 ǀ 36.3	46.5 ǀ 76.7	42.5 ǀ **75.3**
Total	49.3 ǀ **73.8**	68.7 ǀ **76.5**	62.2 ǀ **77.9**	58.0 ǀ **76.0**

Table 5: The detailed experiment results (F1 score) of TAGOP and our model under w/ derivation setting w.r.t. answer types and sources. The left side of the delimiter "ǀ" is the results of TAGOP, and the right is ours.

Settings	BART-base		BART-large		Average Gain	
	EM	F1	EM	F1	EM	F1
w/ derivation						
UniRPG	**67.5**	**74.8**	**70.2**	**77.9**		
w/o constraints	65.2	73.7	68.5	76.8	2.0 ↓	1.1 ↓
w/o structure	63.1	70.8	69.6	77.4	2.5 ↓	2.2 ↓
w/o derivation						
UniRPG	**66.1**	**73.4**	**69.1**	**76.4**		
w/o constraints	64.9	72.3	67.8	75.0	1.2 ↓	1.2 ↓
w/o re-weight	63.7	71.1	67.5	75.1	2.0 ↓	1.8 ↓
w/o structure	62.2	69.1	64.5	72.3	4.2 ↓	4.2 ↓

Table 6: Ablation results on the development set of the TAT-QA dataset.

Results on DROP We verify the effectiveness of UniRPG over text based on the DROP dataset. In this setting, unlike reasoning over table and text, UniRPG does not need to consider some operations tailored for table and some non-involved reasoning types, such as CELL, CELL_VALUE and TIMES/DIV. As shown in Table 4, UniRPG also achieves promising performance on the textual dataset DROP with 77.08 EM and 80.42 F1 scores, which means UniRPG is not only suitable for reasoning over hybrid data but also feasible for text.

4.5 Ablation Study

We investigate the effect of decoding constraints, modeling of table structure, and re-weight strategy for noise reduction in training on the development set of the TAT-QA dataset. As shown in Table 6, the performance declines by 2.0 EM/1.1 F1 and 1.2 EM/1.2 F1 on average under w/o and w/ derivation settings when removing the decoding constraints (w/o constraints). It demonstrates that constraint decoding can significantly reduce errors caused by illegal programs. Moreover, if removing the modeling of table structure, the performance declines by 2.5 EM/2.2 F1 and 4.2 EM/4.2 F1 on average under w/ and w/o derivation settings, respectively. It indicates that modeling table structure significantly contributes to the performance of our
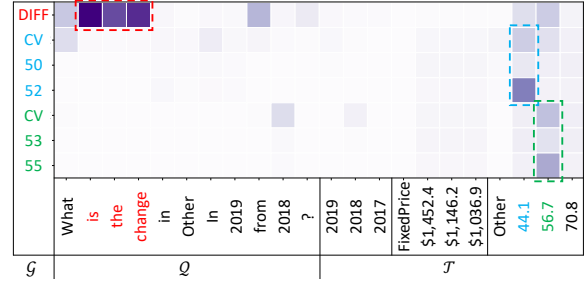

Figure 3: The visualization of the attention score in the last encode layer. The horizontal axis represents the token sequence of the question and table. The vertical axis represents decoded program units at different time steps. Darker color means a higher attention score.
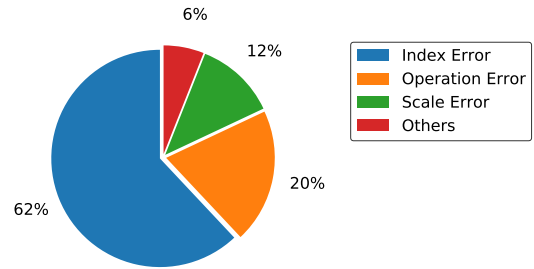

Figure 4: The error distribution of our model.

models. Additionally, we explore the influence of the reweighted mechanism for pseudo programs under the w/o derivation setting. Table 6 shows the re-weight strategy effectively improves the performance by 2.0 EM and 1.8 F1 on average, which treats each type of pseudo program fairly and alleviates negative impact of noise in training.

5 Discussion

Interpretability Analysis As shown in Figure 7-9, our model generates fine-grained programs that intuitively demonstrate the discrete reasoning process. Furthermore, we visually investigate how the neural programmer decodes programs based on questions and hybrid knowledge resources. Figure 3 displays the cross-attention score distribution from the last layer of the decoder based on a random sample. The horizontal axis is the input sequence $\mathcal{S}$, and the vertical axis represents the decoded program units at different time steps. We observe that cross-attention visualization reflects the underlying relationships between the question prefix "*What is the change*" and operation DIFF. Moreover, CV correctly extracts the two numbers 44.1 and 56.7 from the table.

Error Analysis   We analyze our error predictions on the dev set of TAT-QA and observe the error instances fall into the following four categories. The first category is that atomic operations extract wrong text pieces or numbers due to index error, accounting for 62%. We think it is intractable to derive the start and end index from the entire input sequence precisely, which typically has hundreds of tokens. The second category is caused by operation errors when decoding programs and accounts for 20%. We speculate that this error stems from some questions being inherently confusing. For example, given a question *"What is the average USD-EUR exchange rate in FY 2019?"*, the programmer wrongly predicts the operation AVG to solve the question since the keyword *average* appears in the question. However, the correct annotated derivation is "70.07/80.82", i.e., DIV is the proper operation. The remaining errors come from the scale prediction and others, which account for 18%. In summary, more accurately extracting text pieces and numbers from table and text is essential to further improve our model.

Influence of Program Scale   For each training instance in TAT-QA, there are 8 pseudo programs to derive answers correctly on average under the w/o derivation setting. To explore the influence of the program scale, we respectively train models with Top N programs of each instance and all the programs, where $N = 1, 2, ..., 12$. Figure 5 displays the F1 curve of our models on the development set of TAT-QA with respect to the scale of programs N. The F1 score increases with the program scale, then shows a relatively stable performance, reaching a peak when $N = 10$. It indicates the initial increase of F1 is due to gradually sufficient training data, and our model shows good robustness as the noisy data increases.

6   Case Study

We compare our UniRPG with the baseline TAGOP on 50 instances randomly sampled from the development set of TAT-QA, where TAGOP predicts wrongly, but our model predicts correctly. UniRPG has advantages in the following three aspects compared with TAGOP. (1) Capture of argument order. TAGOP introduces a 2-classifier to learn the argument order with a one-hot order label, ignoring the modeling of the interaction between two arguments. Our framework can naturally learn the parameter order of operations because program se-
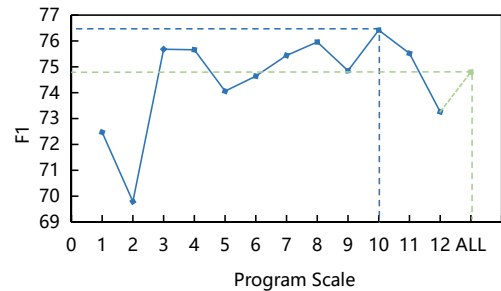

Figure 5: The influence of program scale. ALL means training with all the pseudo programs.

quences are inherently ordered. As shown in example (a), our methods can avoid 60% error instances of TAGOP (Zhu et al., 2021) due to the wrong argument order. (2) Evidence detection. TAGOP extracts evidence by classifying each token of the input and then predicts answers based on the evidence, which is prone to accumulating errors. We employ a copy mechanism to extract content from the input, which only needs to predict the start and end tokens. Example (b) shows our methods reduce 20% error predictions caused by inaccurate evidence compared with TAGOP. (3) Compositional reasoning. As shown in example (c), our model has better scalability that can perform compositional with multiple operations compared to TAGOP.

7   Related Work

Hybrid QA   Hybrid QA aims to derive answers to given questions with heterogeneous knowledge, such as table/knowledge graph and text. Many challenging datasets and promising approaches (Asadifar et al., 2018; Chen et al., 2020b,c; Zhu et al., 2021; Zhong et al., 2022; Wang et al., 2022; Lei et al., 2022) are recently proposed to stimulate the progress of this topic. Chen et al. (2020c) proposed HybridQA, in which each question is aligned with a Wikipedia table and multiple paragraphs linked with cells in the table. Chen et al. (2020b) proposed OTT-QA, a large-scale open table-and-text QA dataset that requires retrieval tables and text to derive answers. Unlike HybridQA and OTT-QA, which study multi-hop reasoning on table and text, we study TAT-QA (Zhu et al., 2021), a large-scale QA dataset requiring discrete reasoning over hybrid data. FinQA (Chen et al., 2021) is a recent work similar to ours, aiming to address the task of numerical reasoning over financial data. Compared to it, UniRPG supports more reasoning capabilities, such as SPAN/CELL, which can perform extraction from paragraphs/tables but is not limited

Example (a):

	2019	2018
Balance at the beginning of the fiscal year	1264	1626
Decreases based on positions taken in prior years	119	304
Balance at the end of the fiscal year	1258	1264

p_1: ... the Company has U.S ...
p_2: A reconciliation of the ...
p_3: The entire amount reflected ...
Q: What is the change in the beginning balance between fiscal years 2019 and 2018?
A: -362
D: {Solution: 1264-1626, Table:[(1,2), (3,2)]}

TAGOP:
Evidence: 1264, 1626
OP: Differece
Order: 1
A: 362

UniRPG:
G: DIFF(CV(29, 32) CV(33, 36))
A: -362

Example (b):

	2019	2018
Impairment charges	94.2	394.0
Net losses on sales or disposals of assets	45.1	85.6
Other operating expenses	27.0	33.7

p_1: American Tower and ...
p_2: Other operating expense ...
p_3: Other operating expenses ...
Q: How many expenses segments in 2018 were below \$100 million?
A: 2
D: {Table: [(2,0), (3,0)]}

TAGOP:
Evidence: Impairment charges, Net losses on sales or disposals of assets, Other operating expenses
OP: Count Order: N/A A: 3

UniRPG:
G: COUNT(CELL(41, 49) CELL(59, 61))
A: 2

Example (c):

	2019	2018	2017
Defined contribution schemes	166	178	192
Defined benefit schemes	57	44	20
Total amount charged to income statement	223	222	212

p_1: 24. Post employment benefits
p_2: The Group operates a...
p_3: Accounting policies ...
Q: What is the difference between 2019 average defined contribution schemes and 2019 avere defined benefit schemes? A: 121.5
D: {Solution: (166+178)/2 - (57+44)/2, Table:[(1,1), (1,2), (2,1),(2,2)]}

TAGOP:
Evidence: 166.0, 57.0
OP: Differece
Order: 0
A: -109.0

UniRPG:
G: DIFF(AVG(CV(44, 44), CV(45,45)), AVG(CV(51, 51), CV(52 ,52)))
A: 121.5

Figure 6: Three examples display the prediction results of baseline TAGOP and our model. In TAGOP, order=N/A means that the current operation does not require an order, and order is 0 implies that the order of the two numbers is the same as the order in the input sequence; otherwise, it is 1.

to the arithmetic computing problem. This enables UniRPG to be available for more discrete reasoning tasks. Moreover, the atomic and higher-order operation framework provides a unified paradigm to define operations, which makes UniRPG convenient to be adapted to other discrete reasoning tasks. In addition, we provide a weak-supervised approach for discrete reasoning over hybrid context based on question-answer pairs, largely reducing the cost of program annotations.

Discrete reasoning for QA Dua et al. (2019) proposed DROP, a large-scale RC dataset to investigate discrete reasoning over text. Many customized models are proposed to tackle this task, and these fall into two dominant directions, the first direction is to devise specific modules for each types of question, such as NAQANET (Dua et al., 2019), Num-Net (Ran et al., 2019), QDGAT (Chen et al., 2020a), EviDR (Zhou et al., 2021) and OPERA (Zhou et al., 2022a). The second direction is to predict programs that would solve the questions (Andor et al., 2019; Gupta et al., 2020; Chen et al., 2020d). It is similar to ours, but our model supports discrete reasoning over hybrid data.

8 Conclusion

In this work, we propose to perform discrete reasoning over table and text as program generation. Our framework consists of a neural programmer to parse questions into executable programs and a symbolic program executor to derive answers based on decoded programs. Furthermore, we design a distant supervision method to automatically construct pseudo programs to alleviate the cost of manual annotation. Comprehensive experiments on the TAT-QA and DROP datasets demonstrate the effectiveness and generality of our model under both full and distant supervision.

Limitations

In this section, we discuss the limitations of this work. First, we heuristically search for possible programs based on manually-designed patterns under weak supervision. It may require expanding and modifying the patterns to derive possible programs when adapted to other tasks. However, the cost of expanding and modifying the patterns is much less than manually labeling programs(derivation). Second, we consider all the possible pseudo programs for each instance and ignore explicitly modeling to choose the correct one from noisy pseudo programs. Experiment results show that the distantly supervised model based on automatically constructed pseudo-programs only lags behind fully supervised model 1.0 EM and 1.4 F1 on the test set. In our experiments, modeling selection of the correct programs from noisy pseudo programs has limited room for improvement, but it remains a valuable topic to explore.

References

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. 2019. Giving BERT a calculator: Finding operations and arguments with reading comprehension. 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 5947–5952, Hong Kong, China. Association for Computational Linguistics.

Somayeh Asadifar, Mohsen Kahani, and Saeedeh Shekarpour. 2018. Hcqa: Hybrid and complex question answering on textual corpus and knowledge graph. *ArXiv preprint*, abs/1811.10986.

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2503–2514, Osaka, Japan.

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 967–976.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. Lgesql: Line graph enhanced text-to-sql model with mixed local and nonlocal relations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 2541–2555.

Kunlong Chen, Weidi Xu, Xingyi Cheng, Zou Xiaochuan, Yuyu Zhang, Le Song, Taifeng Wang, Yuan Qi, and Wei Chu. 2020a. Question directed graph attention network for numerical reasoning over text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6759–6768, Online. Association for Computational Linguistics.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Yang Wang, and William W Cohen. 2020b. Open question answering over tables and text. In *International Conference on Learning Representations*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020c. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036, Online. Association for Computational Linguistics.

Xinyun Chen, Chen Liang, Adams Wei Yu, Denny Zhou, Dawn Song, and Quoc V. Le. 2020d. Neural symbolic reader: Scalable integration of distributed and symbolic representations for reading comprehension. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, et al. 2021. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R Woodward, Jinxia Xie, and Pengsheng Huang. 2021. Towards robustness of text-to-sql models against synonym substitution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 2505–2515.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 946–958, Online. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. 2020. Neural module networks for reasoning over text. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. 2019. A multi-type multi-span network for reading comprehension that requires discrete reasoning. 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 1596–1606, Hong Kong, China. Association for Computational Linguistics.

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

Fangyu Lei, Shizhu He, Xiang Li, Jun Zhao, and Kang Liu. 2022. Answering numerical reasoning questions in table-text hybrid contents with graph-based encoder and tree-based decoder. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1379–1390.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022. TAPEX: table pre-training via learning a neural SQL executor. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

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*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. NumNet: Machine reading comprehension with numerical reasoning. 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 2474–2484, Hong Kong, China. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Yingyao Wang, Junwei Bao, Chaoqun Duan, Youzheng Wu, Xiaodong He, and Tiejun Zhao. 2022. MuGER^2: Multi-granularity evidence retrieval and reasoning for hybrid question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, Online. Association for Computational Linguistics.

Hongzhi Zhang, Yingyao Wang, Sirui Wang, Xuezhi Cao, Fuzheng Zhang, and Zhongyuan Wang. 2020. Table fact verification with structure-aware transformer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1624–1629.

Wanjun Zhong, Junjie Huang, Qian Liu, Ming Zhou, Jiahai Wang, Jian Yin, and Nan Duan. 2022. Reasoning over hybrid chain for table-and-text open domain question answering. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 4531–4537. ijcai.org.

Yongwei Zhou, Junwei Bao, Chaoqun Duan, Haipeng Sun, Jiahui Liang, Yifan Wang, Jing Zhao, Youzheng Wu, Xiaodong He, and Tiejun Zhao. 2022a. OPERA: Operation-pivoted discrete reasoning over text. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1655–1666, Seattle, United States. Association for Computational Linguistics.

Yongwei Zhou, Junwei Bao, Haipeng Sun, Jiahui Liang, Youzheng Wu, Xiaodong He, Bowen Zhou, and Tiejun Zhao. 2021. Evidr: Evidence-emphasized discrete reasoning for reasoning machine reading comprehension. In *The 10th CCF International Conference on Natural Language Processing and Chinese Computing, NLPCC 2021, Qingdao, China*, volume 13028, pages 439–452. Springer.

Yuxuan Zhou, Xien Liu, Kaiyin Zhou, and Ji Wu. 2022b. Table-based fact verification with self-adaptive mixture of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 139–149.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 3277–3287, Online. Association for Computational Linguistics.

A Appendix

A.1 Implementation Details

In our experiments, we employ BART (Lewis et al., 2020) as the initialization parameters of UniRPG and optimize it with AdamW (Loshchilov and Hutter, 2019). Table 7 displays all the hyperparameters for the TAT-QA and DROP datasets during training and inference. Our experiments are all performed on 1 NVIDIA A100 GPU.

Parameters	TAT-QA		DROP
	base	large	large
Learning Rate	1e-4	1e-4	1e-4
Batch Size	128	256	256
Weight Decay	0.01	0.01	0.01
Gradient Clipping	5	5	5
Warmup Steps	0.01	0.01	0.01
Warmup	Linear	Linear	Linear
Maximum Decoding Steps	50	50	50
Beam Size	4	4	4
Training Epochs	30	30	20
Lower Layers	3	4	-
λ	0.3	0.3	-

Table 7: Hyperparameters of UniRPG during training and inference on the TAT-QA and DROP dataset.

A.2 Decoding Constraints

Table 8 displays three categories of decoding constraints to ensure program legality, including index, type, and compositional constraints. We implement these constraints by proposing a decoding mask to filter illegal program units at each step during inference.

Constraint Type / Constraints
Index Constraints
The end index must equal to or be larger than the start index.
A SPAN answer satisfies a length limitation constrained by the difference between the end and start index.
We constrain the index to ensure those span items in a MULTI_SPANS answer are different.
Type Constraints
CELL, CELL_VALUE, SPAN and VALUE should return a correct type of the result, such as CELL_VALUE should return a number from table rather than paragraphs.
Compositional Constraints
The first decoded program unit (except the start identifier $\langle s \rangle$) should be an atomic operation or higher-order operation, except KV.
Operations must take correct types of operation as arguments that is exhaustively demonstrated in Table 1.
Set a maximum number of operation augments, such as no more than three numbers are involved in AVG.

Table 8: Three types of decoding constraints. Each item starting with the character # means a kind of constraint.

A.3 Templates for searching pseudo programs

Table 9 displays the template library for searching pseudo programs under the w/o derivation setting. F_1 and F_2 is anyone of $\{SUM, DIFF, TIMES, DIV\}$. CV is the abbreviation of operation CELL_VALUE. C means a constant defined in Table 1. Note that each operation with a numeric subscript means an instance of this operation.

Categories / Program Templates
Text/Number Extraction
CELL(s, e)
CV(s, e)
SPAN(s, e)
VALUE(s, e)
Arithmetic computing
AVG(CV$_1$/VALUE$_1$, CV$_2$/VALUE$_2$)
AVG(CV$_1$/VALUE$_1$, CV$_2$/VALUE$_2$, CV$_3$/VALUE$_3$)
DIFF(AVG$_1$, AVG$_2$)
CHANGE_R(CV$_1$/VALUE$_1$, CV$_2$/VALUE$_2$)
DIFF(CHANGE_R$_1$, CHANGE_R$_2$)
F$_1$(CV$_1$/VALUE$_1$/CONST, CV$_2$/VALUE$_2$/CONST)
F$_1$(F$_2$(CV$_1$/VALUE$_1$/C, CV$_2$/VALUE$_2$/C), CV$_3$/VALUE$_3$/C))
Comparison Sorting
ARGMAX(KV$_1$, ..., KV$_m$)
ARGMIN(KV$_1$, ..., KV$_m$)
Multi-spans Extraction
MULTI_SPANS(CELL$_1$/SPAN$_1$, ..., CELL$_m$/SPAN$_m$)
Counting
COUNT(CELL$_1$/SPAN$_1$, ..., CELL$_m$/SPAN$_m$)

Table 9: Template library $\mathcal{B}$ for searching pseudo programs under the w/o derivation setting.

A.4 Examples

Figure 7- 9 displays several instances, decoded programs and predictions. Take the first instance in Figure 7 as example, UniRPG translates the question "*In which year is the amount of total sales the largest?*" as the program $\mathcal{G} = \text{ARGMAX}(\text{KV}(\text{CELL}(27, 27), \text{CV}(62, 67)),$ $\text{KV}(\text{CELL}(28, 28), \text{CV}(68, 73)),$ $\text{KV}(\text{CELL}(29, 29), \text{CV}(74, 79)))$. Instead of only outputting a string "2019", we decode the program $\mathcal{G}$ that selects the key with largest value from the key-value pairs $\{\langle 2019, 1496.5 \rangle, \langle 2018, 1202.9 \rangle, \langle 2017, 1107.7 \rangle\}$ to derive the answer. Intuitively, our model generates fine-grained programs, illustrating the discrete reasoning process over table and text.

	2019	2018	2017
Fixed Price	1,452.4	1,146.2	1,036.9
Other	44.1	56.7	70.8
Total sales	1,496.5	1,202.9	1,107.7

p_1: Sales by Contract Type: Substantially all of our contracts are fixed-price type contracts. Sales included in Other contract types represent cost plus and time and material type contracts.

p_2: ...we are paid our allowable incurred costs plus a profit which can be fixed or variable depending on the contract's fee arrangement up to predetermined funding levels determined by the customer. ...

Q: In which year is the amount of total sales the largest?
A: 2019
Prediction Program:
G: ARGMAX(KV(CELL(27, 27), CV(62,67)), KV(CELL(28,28), CV(68,73)), KV(CELL(29,29), CV(74,79))) ⟹ ARGMAX(<2019, 1496.5>,<2018, 1,202.9>,<2017, 1,107.7>)
Prediction Answer: A = 2019

Q: What is the percentage change in Other in 2019 from 2018?
A: -22.22
Prediction Program:
G=CHANGE_R(CV(51,53), CV(54,56)) ⟹ (44.1-56.7)/56.7
Prediction Answer: A = -22.22

Figure 7: Examples in the development of TAT-QA dataset.

	2019	2018	2017
Defined contribution schemes	166	178	192
Defined benefit schemes	57	44	20
Total amount charged to income statement (note 23)	223	222	212

p_1: 24. Post employment benefits

p_2 : ...The Group operates a number of defined benefit and defined contribution pension plans for our employees. The Group's largest defined benefit scheme is in the UK...

p_m: ...On 19 October 2017, the 31 March 2016 triennial actuarial valuation for the Vodafone Section and CWW Section of the Vodafone UK plan...

Q: What is the 2019 average defined contribution schemes?
A: 172
Prediction Program:
G =AVG(CV(36,36), CV(37, 37)) ⟹ (166+178)/2
Prediction Answer: A = 172

Q: What is the difference between 2019 average defined contribution schemes and 2019 average defined benefit schemes?
A: 121.5
Prediction Program:
G=DIFF(AVG(CV(44,44),CV(45,45)), AVG(CV(51,51), CV(52,52))) ⟹ [(166+178)/2] - [(57+44)/2]
Prediction Answer: A = 121.5

Figure 8: Examples in the development of TAT-QA dataset.

	2019	2018	2017
Impairment charges	$94.2	$394.0	$211.4
Net losses on sales or disposals of assets	45.1	85.6	32.8
Other operating expenses	27.0	33.7	11.8
Total Other operating expenses	$166.3	$513.3	$256.0

p_1: AMERICAN TOWER CORPORATION AND SUBSIDIARIES NOTES TO CONSOLIDATED FINANCIAL STATEMENTS...

p_2 : 17. OTHER OPERATING EXPENSE

p_m: ...other assets and miscellaneous items. Other operating expenses includes acquisition-related costs and integration costs...

Q: How many expenses segments in 2018 were below $100 million?

$\mathcal{A}$: 2

Predicted Program: $\mathcal{G}$ = COUNT(CELL(41, 49), CELL(59,61))

⟹ COUNT(Net losses on sales or disposals of assets, Other operating expenses)

Predicted Answer: $\mathcal{A}$= 2

	2019	2018
Operating loss carryforward	73,260	57,768
Net deferred tax asset before valuation allowance	19,911	15,916
Valuation allowance	(19,911)	(15,916)
Net deferred tax asset	795	772

p_1: NOTE 13 - TAXES ON INCOME

p_2 : B. Deferred income taxes

p_3 : ...Deferred income taxes reflect the net tax effects of temporary differences between the carrying amounts of assets and liabilities for financial reporting purposes and the amounts used for income tax purposes. Significant components of the Company's deferred tax assets are as follows:

p_4: ... it is more likely than not that the deferred tax regarding the loss carryforward and other temporary differences will not be realized in the foreseeable future.

Q: What was the operating loss carryforward amount in 2019 and 2018 respectively?

$\mathcal{A}$: {73,260, 57,768}

Predicted Program: $\mathcal{G}$ = MULTI_SPAN(CELL(35,37), CELL(38 40))

⟹ MULTI_SPAN(73,260, 57,768)

Predicted Answer: $\mathcal{A}$={73,260, 57,768}

Figure 9: Examples in the development of TAT-QA dataset.