

Answering Numerical Reasoning Questions in Table-Text Hybrid Contents with Graph-based Encoder and Tree-based Decoder

Fangyu Lei^{1,2}, Shizhu He^{1,2}, Xiang Li^{1,2}, Jun Zhao^{1,2}, Kang Liu^{1,2,3}

¹National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, China

²School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China

³Beijing Academy of Artificial Intelligence, Beijing, 100084, China

{leifangyu2022, lixiang2022}@ia.ac.cn

{shizhu.he, jzhao, kliu}@nlpr.ia.ac.cn

Abstract

In the real-world question answering scenarios, hybrid form combining both tabular and textual contents has attracted more and more attention, among which numerical reasoning problem is one of the most typical and challenging problems. Existing methods usually adopt encoder-decoder framework to represent hybrid contents and generate answers. However, it can not capture the rich relationship among numerical value, table schema, and text information on the encoder side. The decoder uses a simple predefined operator classifier which is not flexible enough to handle numerical reasoning processes with diverse expressions. To address these problems, this paper proposes a **Relational Graph enhanced Hybrid table-text Numerical reasoning model with Tree decoder (RegHNT)**. It models the numerical question answering over table-text hybrid contents as an expression tree generation task. Moreover, we propose a novel relational graph modeling method, which models alignment between questions, tables, and paragraphs. We validated our model on the publicly available table-text hybrid QA benchmark (TAT-QA). The proposed RegHNT significantly outperform the baseline model and achieve state-of-the-art results¹ (2022-05-05).

1 Introduction

Question Answering (QA) is an important task of natural language processing (NLP), which is often used to assess the intelligence of an agent. QA systems use various types of knowledge to answer natural language questions. Earlier approaches independently utilized structured data such as tables (Pasupat and Liang, 2015; Yu et al., 2018), knowledge bases (Yih et al., 2016; Talmor and Berant, 2018) or unstructured data such as plain texts (Rajpurkar et al., 2016). In fact, real-world QA systems often need to fuse different data

resources with diverse types in answering complex questions. Therefore, in recent years, the hybrid form of question answering over tables and texts (TextTableQA) has attracted more and more attention (Chen et al., 2020a,b, 2021).

There are two major question types for TextTableQA. The first is the fact reasoning question, whose answer is usually a span from the table or linked paragraphs, such as the contents in Wikipedia (Chen et al., 2020a,b). The second is the numerical reasoning question, which usually aims to use the contents of tables and texts for numerical calculation (Zhu et al., 2021; Chen et al., 2021). Most previous work focuses on the first type, while the numerical reasoning questions have been seldom addressed. The existing datasets such as WikiTableQuestions (Pasupat and Liang, 2015) and DROP (Dua et al., 2019) also contain numerical reasoning questions, but solving them requires only one type of data source. Therefore, this paper mainly focuses on answering numerical reasoning questions, especially for those complex questions across texts and tables.

To explore the application of numerical reasoning questions in hybrid contents. Zhu et al. (2021) proposed a hybrid text-table dataset TAT-QA, dedicated to fusing the tabular and textual contents to answer numerical reasoning questions. As shown in Figure 1, for the question “*What is the ratio of compensation expense related...?*”, one needs to get the numerical value, i.e. “0.41”, “278.29” from the table, and “109.7” from the text. Then we need to generate the corresponding numerical expression “ $109.7 / (0.41 \times 278.29)$ ”. To solve such a numerical question, we need to identify the describing texts near the table and understand the contents of the table and the paragraphs.

Previous method (Zhu et al., 2021) regarded this problem as a sequence tagging task. They predefine aggregation operators and use a slot filling method to predict simple derivation. An auxiliary

¹We openly released the source code and data at <https://github.com/lfy79001/RegHNT>

	Number of shares	Weighted-average grant date fair value
Nonvested at December 31, 2017	0.859	\$ 187.01
Granted	0.410	278.29
Vested	-0.492	204.24
Forfeited	-0.038	191.51
Nonvested at December 31, 2018	0.739	\$ 225.93
Granted	0.321	318.75
Vested	-0.290	209.05
Forfeited	-0.061	225.23
Nonvested at December 31, 2019	0.709	\$ 275.00

Restricted Stock Grants—During 2019 and 2018, the Company granted 0.321 and 0.410 shares, respectively, of restricted stock to certain employee and director participants under its share-based compensation plans. Restricted stock grants generally vest over a period of 1 to 4 years. The Company recorded \$72.5, \$109.7, and \$63.0 of compensation expense related to outstanding shares of restricted stock held by employees and directors during 2019, 2018 and 2017, respectively.

At December 31, 2019, there was \$77.9 of total unrecognized compensation expense related to nonvested awards granted to both employees and directors under the Company's share-based compensation plans.

Question Type	Question	Answer	Answer From	Derivation
Arithmetic	What is the ratio of compensation expense related to outstanding shares of restricted stock during 2018 to the total price of restricted stock shares granted between 2017 and 2018?	0.96	Table Text	$\frac{109.7}{(0.41 * 278.29)}$
Arithmetic	What is the percentage change in the total price of nonvested shares from December 31, 2018, to 2019?	16.78	Table	$\frac{((0.709 * 275.00) - (0.739 * 225.93))}{(0.739 * 225.93)}$

Figure 1: An example of TAT-QA. The solid boxes are tables, and the dotted boxes are the corresponding paragraphs. The bottom table shows two complex questions that cannot be solved by the previous method. The same color marks the source of the answer, while the blue dashed arrow points to the source of the answer. For the table, blue cells are T_{column} , yellow cells are T_{row} and gold cells are T_{time} .

number order classifier is used for operators sensitive to the operation orders. Moreover, for the complex numerical computation problems in Figure 1, the model cannot predict the answer because of the absence of predefined operators. Thus, previous models based on predefined operators have a deficiency of low generalizability and flexibility. In addition, another problem of the method is to model tables and the texts solely and could not aggregate information from different data types. As a result, incorrect answers are usually generated because of the incomprehensive information from a single type of data.

To solve these problems, we propose a novel method to model such table-text hybrid data for the TextTableQA task, especially for the numerical reasoning question type. Specifically, we build a heterogeneous graph to capture the relationship between different data types. And different node types and relation types (intra-relation and inter-relation) are defined, as detailed in Appendix A. We expect the QA model to capture the correlation between the tables and the texts and aggregate them effectively. The model could focus on the whole contents rather than a single data type. Then a tree-based decoder is built. And we expect it could make good use of the different data structures from different data types and select appropriate nodes in the heterogeneous graph. After that, an expression tree and a prefix expression are generated. So the model can generate arbitrary forms of derivation

without the need for predefined slots. It eliminates error propagation in the operator prediction module and improves the flexibility and generalizability of numerical reasoning.

Experimental results on the public benchmark TAT-QA demonstrate that our proposed model RegHNT improves EM values by 20.2% and F_1 values by 20.0% over the baseline. Our main contributions are summarized as follows:

- We design a novel graph construction method to model the information from table-text hybrid data, which effectively captures the correlation between tables and texts.
- We propose a tree structure decoder to solve the numerical reasoning problems. Based on our method, an expression tree and a prefix expression are generated. Our approach can cover arbitrary numerical derivation forms and improve the model's flexibility and generalizability. To our knowledge, this is the first tree-based model for TextTableQA.
- We think our graph-tree framework can be used as a strong baseline for the TextTable numerical reasoning task. Empirical results on the TAT-QA dataset demonstrate that the proposed model is effective, which achieves the state-of-the-art performances².

²Leaderboard of TAT-QA: <https://nextplusplus.github.io/TAT-QA>

2 Problem Definition

We represent a natural language question as $Q = (q_1, q_2, \dots, q_{|Q|})$ with length $|Q|$. Each question is associated with a table T and a paragraph $P = (s_1, s_2, \dots, s_{|P|})$ with the number of sentences $|P|$. The table T consists of several cells $T = \{c_1, c_2, \dots\}$ and each table cell c_i can be further divided into K words $(c_{i1}, c_{i2}, \dots, c_{iK})$. Similarly, each paragraph sentence s_i contains several words $(s_{i1}, s_{i2}, \dots, s_{iL})$ with the sentence length L . Our goal is to generate the ground truth answer through numerical calculation (see Figure 1).

	Arithmetic Type
Answer	16.78
Expression	$[(0.709*275.00)-(0.739*225.93)]/(0.739*225.93)$
Polish Notation	$/ - * 0.709 275.00 * 0.739 225.93 * 0.739 225.93$
Tree	

Figure 2: The expressions of arithmetic questions. We predict an mathematical expression consisting of numeric nodes and operators to get the answer.

The examples of expressions can be seen in Figure 2. For arithmetic questions which need numerical calculation, inspired by math word problem solving (Liu et al., 2019a; Wang et al., 2019a; Zhang et al., 2020), we generate the final mathematical expression E_p , which is the polish notation transformed from the original math expression. E_p can always be represented as a solution expression tree T_e because the preorder traversal result of the tree is the polish notation.

3 Method

In this section, we present in detail the modules of **Relational Graph enhanced Hybrid** table-text **Numerical reasoning** model with **Tree** decoder (**RegHNT**) (see Figure 3). First, a text-table hybrid data modeling approach is proposed. After constructing the graph, we utilize a classic encoder-decoder architecture for predicting answers. It consists of a graph input module, a graph enhanced hidden module, and a tree-based decoder module. Both the graph input module and the graph enhanced hidden module are parts of the encoder,

aiming to map the input heterogeneous graph G into node embeddings $\mathbf{Z} \in \mathbb{R}^{|V| \times d}$, where d is the graph hidden size. The tree-based decoder module is responsible for transforming \mathbf{Z} into the target T_e .

3.1 Graph Construction

The entire input heterogeneous graph $G = (V, R)$ consists of all types of nodes, that is $V = Q \cup T \cup P$ with the number of nodes $|V| = |Q| + |T| + \sum_{s_i \in P} |s_i|$, where $|T|$ and $|s_i|$ are the number of table cells and paragraph sentences respectively. For relations, $R = R_I \cup R_C$. R_I denotes intra-relation and R_C denotes inter-relation. Details of the node types and relation types are described in the Appendix A. Finally, we model the question, table and text as a graph.

3.2 Graph Input Module

The graph input module aims to initialize embeddings for both nodes and edges. For edges, the edge features are directly initialized from a parameter matrix. For nodes, we can obtain their representations from a pre-trained language model (PLM) such as RoBERTa (Liu et al., 2019b). We flatten all question words, table cells and paragraph words into a sequence $[\text{CLS}]q_1q_2\dots q_{|Q|}[\text{SEP}]t_1t_2\dots t_{|T|}[\text{SEP}]s_{10}s_{11}\dots s_{|P|}0s_{|P|}1\dots[\text{SEP}]$. s_{i0} is the special sentence token of the sentence s_i . Since each word e of the sequence is tokenized into sub-words, we use **Multi-granularity type aware pooling** to get the node representation x . Details in the Appendix B.

3.3 Relational Graph Enhanced Module

This module aggregates information about the nodes and edges of the heterogeneous graph. It is a stack of L relational graph attention network (RGAT) layers. In each layer l , the RGAT (relational graph attention transformers) (Wang et al., 2020) models the graph G and computes the output representation \mathbf{Z} by:

$$\begin{aligned}
 e_{ij}^{(h)} &= \frac{x_i W_q^{(h)} (x_j W_k^{(h)} + r_{ij}^K)^{(T)}}{\sqrt{d_z/H}} \\
 \alpha_{ij}^{(h)} &= \text{softmax} \left\{ e_{ij}^{(h)} \right\} \\
 z_i^{(h)} &= \sum_{v_j \in \mathcal{N}_i} \alpha_{ij}^{(h)} (x_j W_v^{(h)} + r_{ij}^V)
 \end{aligned} \tag{1}$$

where matrices W_q, W_k, W_v are trainable parameters in self-attention, and \mathcal{N}_i is the receptive field

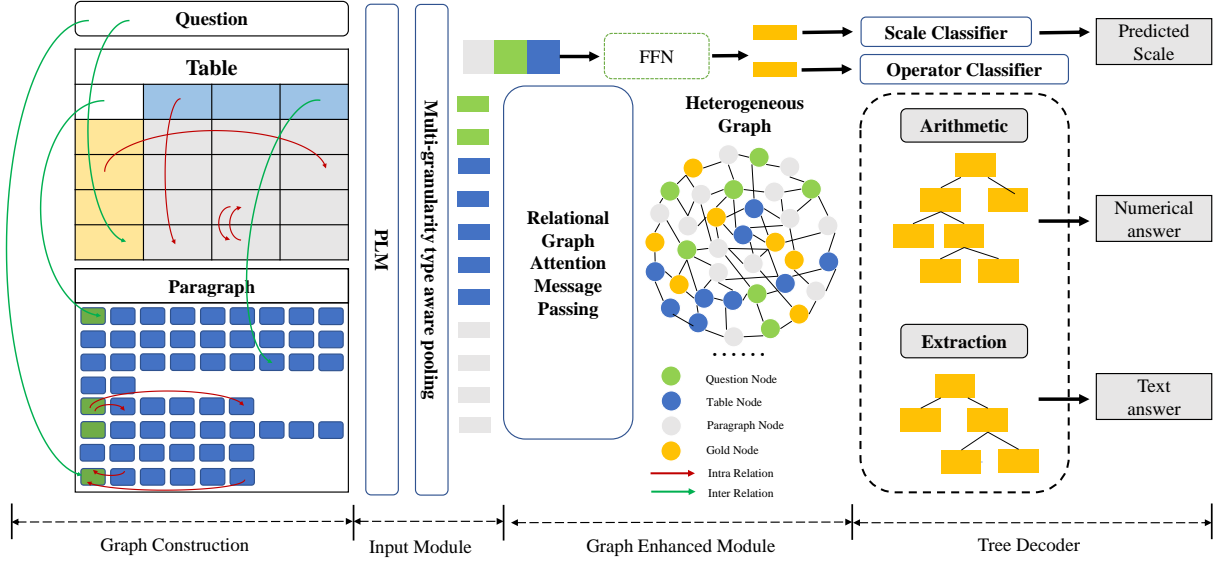


Figure 3: The overall model architecture. The dashed box is the tree-based decoder. Depending on the type of question, two separate trees are constructed to generate the answer.

of node v_i . The output matrices of the final layer L are the desired outputs of the encoder: $Z = Z^L$.

3.4 Tree-based Decoder Module

Inspired by the goal-driven tree structure (GTS) (Xie and Sun, 2019) for solving math word problem, we propose a novel tree-based decoder to construct the calculation expressions for solving text-table numerical reasoning problem. As such, the specialized tree decoder generates an equation following the pre-order traversal ordering. The model takes in question Q , table T , paragraph P and generates a expression tree T_e . Let V_{num} denote numeric values in T and P . Generally, V_{con} denotes constant values $V_{con} = \{1, \text{AVG}\}$, AVG means to average the sum of the previous numbers. V_{op} denotes mathematical operators $V_{op} = \{+, -, \times, \div\}$.

The tree generation process is designed as a pre-order tree traversal (root-left-right). For node y in target T_e , $y \in V_{num} \cup V_{con} \cup V_{op}$. We set V_{num} and V_{con} to be the leaf nodes and V_{op} serve as the internal nodes and must have two child nodes.

The tree structured decoder uses the final graph layer representations z_i as input and generates the target expression in t time steps. At each time step t , let s_t denote the decoding hidden state, c_t denotes the hybrid context state, g_t denotes the generated expressions tree state.

The decoder is a bi-directional GRU (Cho et al., 2014), which updates its states at time step $t + 1$ as

follows:

$$s_{t+1} = \text{BiGRU}([c_t : g_t : E(y_t)], s_t)$$

where $E(y_t)$ is the embedding of token y_t :

$$E(y_t) = \begin{cases} \mathbf{M}_{op}(y_t) & \text{if } y_t \in V_{op} \\ \mathbf{M}_{con}(y_t) & \text{if } y_t \in V_{con} \\ h_{loc(y_t, T, P)}^i & \text{if } y_t \in V_{num} \end{cases} \quad (2)$$

\mathbf{M}_{op} and \mathbf{M}_{con} are two trainable embeddings for operators and constants, respectively. For a numeric value in V_{num} , its token embedding takes the corresponding hidden state $h_{loc(y_t, T, P)}^i$, where $loc(y_t, T, P)$ is the index position of y in table T or paragraph P (Hong et al., 2021).

Inspired by math word problem solving (Wu et al., 2021), the generated expression tree state g_t is calculated as follows:

$$g_{t+1} = \sigma(W_g [g_t : g_{g,p} : g_{g,l} : g_{g,r}]) \quad (3)$$

where σ is a sigmoid function and W_g is a weight matrix. For each generated node, $g_{g,p}$, $g_{g,l}$, $g_{g,r}$ represent the expression tree state of the parent node, left child node, and right child node of the current node, respectively.

The hybrid context state c_t is computed via attention mechanism as follows:

$$\alpha_{ti} = \text{softmax}(\tanh W_h z_i + W_s [s_t : r_t])$$

$$c_t = \sum_{i=1}^m \alpha_{ti} z_i \quad (4)$$

where W_h, W_s are weight matrices. α_{ti} is the attention distribution on the node representations \mathbf{z}_i .

Lastly, the decoder can generate a word from a given vocabulary $V_{op} \cup V_{con}$. It can also generate a number symbol from V_{num} , which is copied a number from the table T or paragraph P . The final distribution is the combination of the generated probability and copy probability:

$$\begin{aligned} p_c &= \sigma(W_z[\mathbf{s}_t : \mathbf{c}_t : \mathbf{r}_t]) \\ P_c(y_t) &= \sum_{y_t=x_i} \alpha_{ti} \\ P_g(y_t) &= \text{softmax}(f([\mathbf{s}_t : \mathbf{c}_t : \mathbf{r}_t])) \\ P(y_t|y_{<t}, X) &= p_c P_c(y_t) + (1 - p_c) P_g(y_t) \end{aligned} \quad (5)$$

Here, $f(\cdot)$ is a perception layer. p_c is the probability that the current word is a number copied from the table or paragraphs.

3.5 Operator and Scale Prediction

In addition, there are two separate tasks in the decoding section: **operator prediction** and **scale prediction**. For arithmetic questions, a right prediction of a numerical answer should include the right number and the correct scale. The scale in the dataset may be *None, Thousand, Million, Billion, and Percent* generally. We focus on arithmetic questions for operator prediction, but there are still non-arithmetic questions (span extraction question) in the dataset. So we classify whether the question is arithmetic or not before decoding. For the extraction questions, as shown in Figure 3, we also model them as trees, as described in Appendix C.

To predict the right aggregation operator and scale, two multi-class classifiers are developed. In particular, we take the vector $\langle \text{CLS} \rangle$ to compute the probability:

$$\begin{aligned} p^{op} &= \text{softmax}(\text{FFN}([\langle \text{CLS} \rangle; h_Q; h_T; h_P])) \\ p^{\text{scale}} &= \text{softmax}(\text{FFN}([\langle \text{CLS} \rangle])) \end{aligned} \quad (6)$$

where h_Q, h_T and h_P are the representations of the question, the table and the paragraphs, respectively, which are obtained by applying an average pooling over the representations of their corresponding tokens. “;” denotes concatenation, and FFN denotes a two-layer feed-forward network with the GELU activation.

3.6 Training

To optimize **RegHNT**, the overall loss is the sum of the loss of the above tasks:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{tree} + \mathcal{L}_{op} + \mathcal{L}_{scale} \\ \mathcal{L}_{tree} &= - \sum_{t=1}^T \log \mathbf{P}(y_t | y_{<t}, Q, T, P) \\ \mathcal{L}_{op} &= \text{NLL}(\log(\mathbf{P}^{op}), \mathbf{G}^{op}) \\ \mathcal{L}_{scale} &= \text{NLL}(\log(\mathbf{P}^{\text{scale}}), \mathbf{G}^{\text{scale}}) \end{aligned} \quad (7)$$

\mathcal{L}_{tree} is the loss function of training the tree-decoder, and we use the cross-entropy loss. \mathcal{L}_{op} and \mathcal{L}_{scale} are the loss functions for operator prediction and scale prediction, respectively, where $\text{NLL}(\cdot)$ is the negative log-likelihood loss. \mathbf{G}^{op} comes from the supporting evidence, which is extracted from the annotated answer and derivation. $\mathbf{G}^{\text{scale}}$ uses the annotated scale of the answer. We add up the three loss functions as the total loss function.

4 Experiments

4.1 Dataset and Evaluation Metrics

TAT-QA (Zhu et al., 2021) is a large-scale, hybrid QA dataset which contains numerical reasoning and span extraction questions. And the contents of TAT-QA include both tabular and textual data from real financial reports. It contains a total of 2,757 hybrid contexts and 16,552 corresponding question-answer pairs. The detailed statistics are shown in Appendix D. The original dataset contains four types of questions: *Span, Multi-Span, Count, Arithmetic*. But in our setup, there are two types of questions: *Span Extraction* and *Arithmetic*. And our work mainly focuses on answering the *Arithmetic* questions.

For evaluation, we adopt the Exact Match (EM) and numeracy-focused F1 score (Dua et al., 2019) to measure the performance of different QA models. All of which are computed using the official evaluation script³. We submit our model to the organizer of the challenge for evaluation. The evaluation detail can be found on the original paper (Zhu et al., 2021).

4.2 Implementation Details

Implementations. Our model is implemented with PyTorch (Paszke et al., 2019), and the graphs are

³<https://github.com/NExTplusplus/tat-qa>

constructed with the library DGL (Wang et al., 2019b). In the graph input module, we use pre-trained language models (PLMs) RoBERTa (Liu et al., 2019b) to obtain the initial representations. During evaluation, we adopt beam search decoding with beam size 3.

Hyper-parameters. In the encoder, the number of GNN layers L is 8, and the number of heads in multi-head attention is 8. For PLMs, we use learning rate $1e-5$ and weight decay rate 0.01. For other model modules, we use a larger learning rate $1e-4$, and a weight decay rate $5e-5$. In the decoder, The recurrent dropout rate (Gal and Ghahramani, 2016) is 0.2 for GRU. The number of heads in multi-head attention is 8 and the dropout rate of features is set to 0.1 in both the encoder and decoder. Throughout the experiments, we use AdamW (Loshchilov and Hutter, 2018) optimizer with a linear warmup scheduler. The warmup ratio of the total training steps is 0.06. The batch size is 48, and the training epoch is 100. The training process may take around 2 days using a single NVIDIA GeForce RTX 3090.

Baselines. We compare with the standard TAGOP (Zhu et al., 2021), which first applies sequence tagging to extract relevant cells from the table and text spans from the paragraphs. We also compare with other advanced models, which can be found on the TAT-QA challenge leaderboard⁴. There are no linked papers to the submissions as yet. We compare our model’s performance on the test split with all of them.

4.3 Main Results

Method	Dev		Test	
	EM	F ₁	EM	F ₁
Human	-	-	84.1	90.8
TAGOP	55.2	62.7	50.1	58.0
LETTER	-	-	56.1	64.3
KIQA	-	-	58.2	67.4
GSReasoner	-	-	67.4	75.5
RegHNT	73.6	81.3	70.3	78.0

Table 1: The performance of different models on dev and test set of TAT-QA. The best results are marked in bold.

The main results on the test set are provided in Table 1. Our model achieves the state-of-the-art results in the publicly available TAT-QA benchmark

⁴<https://nextplusplus.github.io/TAT-QA>

	Table	Text	Table-text
	EM/F ₁	EM/F ₁	EM/F ₁
TAGOP			
Span	56.5/57.8	56.5/57.8	68.2/71.7
Spans	66.3/77.0	19.0/59.1	63.2/76.9
Counting	63.6/63.6	-/-	62.1/62.1
Arithmetic	41.1/41.1	27.3/27.3	46.5/46.5
RegHNT			
Span	68.5/70.0	58.7/83.0	77.0/84.7
Spans	79.5/86.2	23.8/65.3	81.1/90.1
Counting	36.3/36.3	-/-	82.7/82.7
Arithmetic	72.7/72.7	27.3/27.3	77.7/77.7

Table 2: Detailed experimental results of TAGOP and RegHNT w.r.t. answer types and sources on the test set.

and achieves 20% higher on both EM and F₁ compared with the original baseline (TAGOP), which shows that our model can answer more questions with higher accuracy.

The detailed results on the test set are provided in Table 2. For almost all types of questions, the accuracy of RegHNT prediction has been improved. Thanks to the tree decoder for arithmetic questions, the model’s accuracy on this type of questions has been greatly improved, with an overall improvement of almost 30%. For span extraction questions (*Span*, *Spans*, *Counting*), we observe a improvement in the performance of the model as well. From the perspective of answer sources, compared with the original baseline (TAGOP), we focus on "table-text" type questions, both "arithmetic" and "span extraction" type show a performance improvement of about 20%. It demonstrates that our approach works very well in solving table-text hybrid questions.

This paper focuses on the numerical reasoning questions. To verify our model more precisely, we divided the questions into three categories based on the complexity of the derivation, as shown in Table 3. Simple arithmetic has only one operator. Complex arithmetic has multiple operators, usually used to calculate the average and the rate of change. Undefined arithmetic is arithmetic for which no template is defined in TAGOP. The results show that our model significantly improves both simple and complex arithmetic. In particular, our model can solve undefined arithmetic of TAGOP, which offers flexibility and generalizability.

Arithmetic type	%	TAGOP	RegHNT
Simple arithmetic	41.8	45.0	79.3
Complex arithmetic	42.8	60.5	85.3
Undefined arithmetic	15.4	0.0	61.8

Table 3: Exact match value for different types of arithmetic questions.

Technique	EM	F ₁
RegHNT	73.6	81.3
w/o Intra-relations	72.8	80.4
w/o Inter-relations	72.3	79.8
w/o All relations	71.6	78.7
w/o Multi-granularity type aware pooling	73.0	80.7
w/o Tree-decoder (Span extraction)	72.8	80.6
w/o Tree-decoder (All questions)	60.3	69.5

Table 4: Ablation study of different modules.

4.4 Ablation Studies

Effect of Tree Decoder. Although our work focuses on arithmetic questions, to unify the whole model into a graph-tree framework, we transform the span extraction type question into a tree generation problem as well, as mentioned in Section 3.5. We conducted two experiments using the sequence tagging method in TAGOP (Zhu et al., 2021) instead of generating expression trees. As shown in the last two rows of Table 4, one is to use sequence tagging for all questions, and the other is to use sequence tagging only for span extraction questions. When we change the decoder only for extraction questions, the F₁ drops only 0.7%, but when we change the decoder for all questions, the F₁ drops about 11.8%. It shows that the tree decoder is not only tremendously helpful in solving arithmetic questions but also provides a slight improvement in solving span extraction questions.

Effect of Graph Encoder. As shown in the first four rows of Table 4, we show the effectiveness of the proposed graph encoder. Removing the intra-relations reduces the F₁ value by 0.9% and reduces the EM value by 0.8%. Removing the inter-relations reduces F₁ value by 1.5% and reduces the EM value by 1.3%. When we remove all relations (remove graph enhanced module), the F₁ decreases by 2.6%, and the EM value decreases by 2.0%. From the results, it can be clearly confirmed that the graph we built plays an essential role in modeling table-text hybrid data, and it captures the semantic association through the message passing between different data types.

Effect of Subword Pooling Layer. In the graph in-

put module, we used a multi-granularity type aware pooling method. The type classification criteria for word granularity are text and number, and for node granularity are question, table, and paragraph. As shown in the fifth row “w/o Multi-granularity type aware pooling” of Table 4, we eliminate this mechanism and unify the pooling approach for all types and granularities. Experimental results of F₁ dropped by 0.6%, which shows the effectiveness of this type-aware module.

4.5 Scale and Operator Study

Scale Study. Scale prediction is a unique challenge over TAT-QA and very pervasive in the context of finance. After obtaining the scale, the numerical or string prediction is multiplied or concatenated with the corresponding scale as the final prediction to compare with the ground-truth answer, respectively. We compare RegHNT with the baseline model for scale prediction results. The experimental results are shown in Table 5. Our model has significantly improved performance on both the dev and test datasets. To explore the impact of the scale on results, we use the gold scale to predict the answer. As shown in the third row of Table 6, model accuracy will slightly increase to 84.2% when we use the gold scale, which shows that it is necessary to improve the prediction of scale.

Operator Study. For TAT-QA dataset, there are four original answer types: *Span*, *Multi-Span*, *Count*, *Arithmetic*. As Figure 3 shows, we have adapted it into two categories, where the details of the expression tree for the span extraction questions are in the Appendix C. To investigate whether this category setting causes error propagation, we use the gold operator to predict the answer, and the results are shown in Table 6. When we use the gold operator, the EM and F₁ of the model is improved by only 0.1. It suggests, to some extent, that we divide the data into two categories and use tree decoders to generate the answers separately. This approach has no significant impact on performance.

Model	Dev	Test
TAGOP	93.5	92.2
RegHNT	95.3	93.4

Table 5: Scale prediction results of our model and baseline.

Model	EM	F ₁
RegHNT	73.6	81.4
RegHNT + Gold operator	73.7	81.5
RegHNT + Gold scale	76.5	84.2
RegHNT + Gold operator + Gold scale	76.7	84.3

Table 6: The performance of using gold operators and gold scales.

4.6 Case Studies

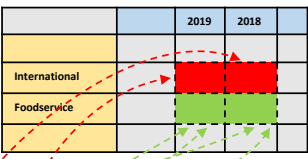
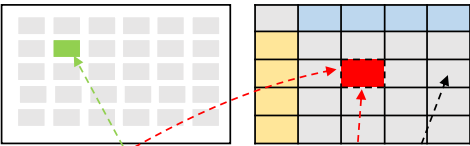
RegHNT	TAGOP
<p>What is the percentage change in total net sales of International and Foodservice from fiscal year 2018 to 2019?</p> 	
$\frac{- + 793.4 \quad 934.2 - + 843.5}{1054.8 \quad + \quad 843.5 \quad 1054.8 -}$	$/ \quad 793.4 \quad 843.5$
$\frac{[(793.4 + 934.2) - (843.5 + 1054.8)]}{(843.5 + 1054.8)}$	$793.4 / 843.5$
<p>What was the average employee termination cost per employee in 2018?</p> 	
$/ \quad 53.0 \quad 1027$	$/ \quad 53.0 \quad 55.5$
$53.0 / 1027$	$53.0 / 55.5$

Figure 4: Two examples of generated expressions by RegHNT and TAGOP(Zhu et al., 2021).

As Figure 4 shows, there are two questions and the prediction results of the models. For the question “What is the percentage change in total net sales...?”, the previous method could not generate complex arithmetic. In contrast, our model can generate the correct prefix expressions, which demonstrates the characteristics and advantages of our tree decoder. For the question “What was the average employee termination...?”, the correct expression is derived from table and text (“53.0” from table and “1027” from paragraph). The results in Figure 4 show that the model often fails to answer correctly when a question requires the use of both tables and text. It focuses only on tables, choose “53.0” and “55.5”. This error type is very common in the previous methods. It shows that our graph encoder can better model the association between tables and texts.

5 Related Work

Table-text hybrid QA is a new task that consists of two main types of work.

Fact Reasoning TextTableQA. Chen et al. (2020b) propose the first table-text hybrid QA dataset. It is the fact reasoning type dataset whose answer is usually a span from the table or linked paragraphs of Wikipedia. The authors supposed HYBRIDER (Chen et al., 2020b), a pipeline approach that divides the prediction process into two phases called linking and reasoning. MITQA (Kumar et al., 2021) achieves SOTA EM result on HybridQA, which is a novel training strategy that works with multiple instances and multiple answers based on weak supervision. DEHG (Feng et al., 2022) propose a document-entity heterogeneous graph network and achieve SOTA F1 score on HybridQA. OTT-QA (Chen et al., 2020a) is a difficult open-domain setting TextTableQA task, which needs retrieval and reading to get the answers. CARP (Zhong et al., 2022) utilizes hybrid chain to model the explicit intermediate reasoning process across table and text for question answering, which achieves SOTA results, but still far from expectations. GeoTSQA (Li et al., 2021) is a multiple choice QA dataset based on geography domain.

Numerical Reasoning TextTableQA. TAT-QA (Zhu et al., 2021) and FinQA (Chen et al., 2021) are the numerical reasoning hybrid dataset which comes from the financial field. Both TAT-HQA (Li et al., 2022) and TAT-DQA (Zhu et al., 2022) are enhanced datasets of TAT-QA, which study TextTableQA in counterfactual condition and multimodal condition, respectively. MULTIHERTT (Zhao et al., 2022) is a challenging dataset, which contains multiple hierarchical tables and longer unstructured text. Unlike HybridQA, which was fact reasoning type questions, TAT-QA focuses explicitly on finance and needs numerical reasoning for question answering over tabular numbers and associated text. They proposed TAGOP model and regarded this problem as a sequence tagging task. It predefined aggregation operators and used a slot filling method to predict simple derivation, lacking generalizability and flexibility. Our model is the first method to generate arithmetic expressions directly for table-text hybrid numerical reasoning QA.

6 Conclusion

This paper proposes a novel method to solve table-text hybrid numerical reasoning problems and achieve good performance. We present a unified framework for addressing the table schema and relative paragraphs. By adopting relation-aware self-attention, the proposed method jointly learns question, table and paragraph representations based on their alignment. At the same time, we offer a tree-based numerical reasoning decoding framework for hybrid data, the first to use this type of method for this task. Our model can serve as a strong baseline for this task. However, our model has not yet been experimented on encyclopedic type questions (Chen et al., 2020b), and we will explore a table-text hybrid QA framework that integrates dealing with factual and numerical reasoning types of questions. For numerical reasoning type questions, we will do further research on TAT-HQA (Li et al., 2022) and TAT-DQA (Zhu et al., 2022).

Acknowledgements

This work is supported by the Natural Key R&D Program of China (No.2022QY0701), the National Natural Science Foundation of China (No.61922085, No.61976211) and the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No. XDA27020200). This research work was supported by the independent research project of National Laboratory of Pattern Recognition (No. Z-2018013), the Youth Innovation Promotion Association CAS, Yunnan Provincial Major Science and Technology Special Plan Projects (No.202202AD080004) and CCF-DiDi GAIA Collaborative Research Funds for Young Scholars.

References

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. Lgesql: Line graph enhanced text-to-sql model with mixed local and non-local relations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2541–2555.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Yang Wang, and William W Cohen. 2020a. 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. 2020b. 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.

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.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

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.

Yue Feng, Zhen Han, Mingming Sun, and Ping Li. 2022. Multi-hop open-domain question answering over structured and unstructured knowledge. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 151–156, Seattle, United States. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. *Advances in neural information processing systems*, 29.

Yining Hong, Qing Li, Daniel Cio, Siyuan Huang, and Song-Chun Zhu. 2021. Learning by fixing: Solving math word problems with weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4959–4967.

Vishwajeet Kumar, Saneem Chemmengath, Yash Gupta, Jaydeep Sen, Samarth Bharadwaj, and Soumen Chakrabarti. 2021. Multi-instance training for question answering across table and linked text. *arXiv preprint arXiv:2112.07337*.

Moxin Li, Fuli Feng, Hanwang Zhang, Xiangnan He, Fengbin Zhu, and Tat-Seng Chua. 2022. Learning to imagine: Integrating counterfactual thinking in neural discrete reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 57–69.

Xiao Li, Yawei Sun, and Gong Cheng. 2021. Tsqa: tabular scenario based question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13297–13305.

- Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. 2019a. Tree-structured decoding for solving math word problems. 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 2370–2379.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.
- Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.
- Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651.
- Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578.
- Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. 2019a. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7144–7151.
- Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. 2019b. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*.
- Qinzhao Wu, Qi Zhang, and Zhongyu Wei. 2021. An edge-enhanced hierarchical graph-to-tree network for math word problem solving. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1473–1482.
- Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *IJCAI*, pages 5299–5305.
- Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.
- Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.
- Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020. Graph-to-tree learning for solving math word problems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3928–3937.
- Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022. **MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600, Dublin, Ireland. Association for Computational Linguistics.
- 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-22*, pages 4531–4537. International Joint Conferences on Artificial Intelligence Organization. Main Track.
- Fengbin Zhu, Wenqiang Lei, Fuli Feng, Chao Wang, Haozhou Zhang, and Tat-Seng Chua. 2022. Towards complex document understanding by discrete reasoning. *arXiv preprint arXiv:2207.11871*.
- 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 (Volume 1: Long Papers)*, pages 3277–3287.

A Relation Detail

Before defining relation types, we'd better introduce the node types more finely. There are three types of nodes, but they can be divided in more detail. Table T has complex structural information. We set three types according to the location of the cell, namely \mathbf{T}_{row} , $\mathbf{T}_{\text{column}}$ and \mathbf{T}_{cell} . Some row headers and column headers are time (e.g. 2019). We set these special cell as \mathbf{T}_{time} . Row headers and column headers are composed of text, while numerical cells are numbers. For the question Q , there is only one node type \mathbf{Q}_{word} . For the paragraph P , there are two node types, \mathbf{P}_{word} represents the common sentence words, $\mathbf{P}_{\text{sentence}}$ represents the special sentence token.

Now there are seven types of nodes, and we establish edges between them, which is mainly divided into intra-relation R_I and inter-relation R_C . The relations are based on the table schema and word matching between resources, and details are described in Table 7.

B Multi-granularity type aware pooling

Since each word e of the sequence is tokenized into sub-words, we need to aggregate them in order to obtain the node representation. We set two granularities (word, node) and three types (question, table, paragraph) aware pooling method.

- Word level: For number word (e.g. 109.7) and text word (e.g. compensation), we use two independent subword attentive pooling module depending on word type to get the type-aware word level representation w .

$$w = \sum_i \text{softmax}_i [\tanh(e_i W_s) v_s^T] e_i$$

- Node Level: For a node, especially a table cell node, which usually consists of multiple words. According to the node source types, we aggregate the word level granularity representation with three different pooling layers and three different BiLSTM to get the node representation x .

$$x = \sum_i \text{softmax}_i [\tanh(w_i W_n) v_n^T] w_i$$

Where v_s , W_s , v_n , W_n are trainable parameters. The attentive pooling layer is inspired by LGESQL (Cao et al., 2021).

C Details of the span extraction question

We also predict an expression tree for span extraction questions to get the answer. The example is shown in Figure 5. Unlike mathematical problems, the leaf node in the span extraction tree represents node ID, while the leaf node in the arithmetic tree represents the numeric number or constant. As for the operator, the operators in the arithmetic tree are “+”, “-”, “×”, “÷”. But for the span extraction tree, we define three operators, “+”, “×”, “C”. “+” represents the splicing of two discontinuous spans, corresponding to the *multi-span* in the original dataset. “×” means taking the operator’s left and right sides as the starting and ending nodes and selecting all nodes in the middle of the two nodes as a span. “C” is the same as “+”, but it counts the number of spans instead of slicing them.

	Span Extraction Type
Answer	integration and transformation-related expenses severance and retention compensation expenses transaction-related expense
Expression	7 + 10 + 44 * 47
Polish Notation	+ + 7 10 * 44 47
Tree	

Figure 5: Expressions for span extraction questions. We also predict an expression tree to get the answer. “44×47” means to select all words between nodes ID 44 and nodes ID 47, “+” means that the span on both sides is the answer.

D Details of the TAT-QA

The specific data analysis of the dataset is shown in Table 8 and Table 9.

Intra-Relation			
Source x	Source y	Relation	Description
T_{row}	T_{cell}	CONTAIN	x is row head of y .
T_{column}	T_{cell}	CONTAIN	x is column head of y .
T_{time}	T_{cell}	CONTAIN	x is row/column head of y , x is time.
P_{sentence}	P_{word}	CONTAIN	x is the sentence token which contain y .
Q_{word}	Q_{word}	DISTANCE+1	y is the next word of x .
P_{word}	P_{word}	DISTANCE+1	y is the next word of x .
T_{cell}	T_{cell}	SAME ROW	x and y are in the same row.
T_{cell}	T_{cell}	SAME COLUMN	x and y are in the same column.
Inter-Relation			
Q_{Word}	T_{row}	PARTIALMATCH EXACTMATCH	x is part of y , but the entire question does not contain y . x is part of y , and y is a span of the entire question.
P_{sentence}	Q_{word}	CONTAIN	x is the sentence token which contain y .
P_{sentence}	T_{row}	CONTAIN	x is the sentence token which contain y .
P_{word}	T_{row}	PARTIALMATCH EXACTMATCH	x is part of y , but the entire sentence does not contain y . x is part of y , and y is a span of the entire question.
P_{word}	Q_{word}	SAME	x and y are the same words.

Table 7: The checklist of all relations in our RegHNT. All the above relations are asymmetric. We show only one direction, and the opposite direction can be easily inferred.

	Table	Text	Table-text	Total
Span	1,801	3,496	1,842	7,139
Spans	777	258	1,037	2,072
Counting	106	5	266	377
Arithmetic	4,747	143	2,074	6,964
Total	7,431	3,902	5,219	16,552

Table 8: Question distribution regarding different answer types and sources in TAT-QA.

Statistic	Train	Dev	Test
# of hybrid contexts	2,201	278	278
# of questions	13,215	1,668	1,669
Avg. rows / table	9.4	9.7	9.3
Avg. cols / table	4.0	3.9	4.0
Avg. paragraphs / table	4.8	4.9	4.6
Avg. paragraph len [words]	43.6	44.8	42.6
Avg. question len [words]	12.5	12.4	12.4
Avg. answer len [words]	4.1	4.1	4.3

Table 9: Basic statistics of each split in TAT-QA.