HG^3: An Information Augmentation Retriever-Generator Model for Financial Numerical Reasoning

Anonymous ACL submission

Abstract

The NLP research community widely recognizes numerical reasoning as a core competency, critical for constructing logically sound solutions to mathematical queries grounded in contextual evidence. Most existing methods are based on the retriever-generator model. However, this two-stage method still lose important information, especially in complex scenarios where longer text and tabular reasoning are mixed. To solve these problems, we propose a **H**eterogeneous **G**raph **G**aussian **G**eneration model (**HG**^{3}), which improves the ability of retriever-generator model to retrieve key facts and generate correct answers from the perspective of information augmentation. In the retrieval stage, we propose using heterogeneous graphs to model the relationships between documents and tables. This approach allows us to capture the structural attributes of tables, thereby enhancing the ability of retrieving critical facts. In the generator stage, we propose a Gaussian process random function to introduce context-aware variations into the encoder, in this way, we can generate high-quality text with key facts as the core by enriching contextual representation learning of the model. Experimental results on the FinQA and ConvFinQA demonstrate the effectiveness of HG^3, which outperforms all the baselines. The code and datasets of this work will be open-sourced after acceptance.

1 Introduction

Numerical reasoning is a fundamental task of NLP technology, which centers on key factual information (evidence) and tests the model's ability to: 1) extract key facts, and 2) generate answers from key facts to solve arithmetic problems. With the rise of large language models (LLMs) like ChatGPT and GPT-4(Adesso, 2022), researchers are beginning to explore more challenging numerical reasoning tasks, such as long-form numerical reasoning. New benchmarks have been introduced to include longer textual data and more complex data formats, which presents significant challenges. For instance, financial numerical reasoning datasets now require models to answer financial analysis questions based on both table and textual data, as seen in datasets like FinQA(Chen et al., 2021), ConvFinQA(Chen et al., 2022), and TAT-QA(Zhu et al., 2021). This is illustrated in Figure 1.

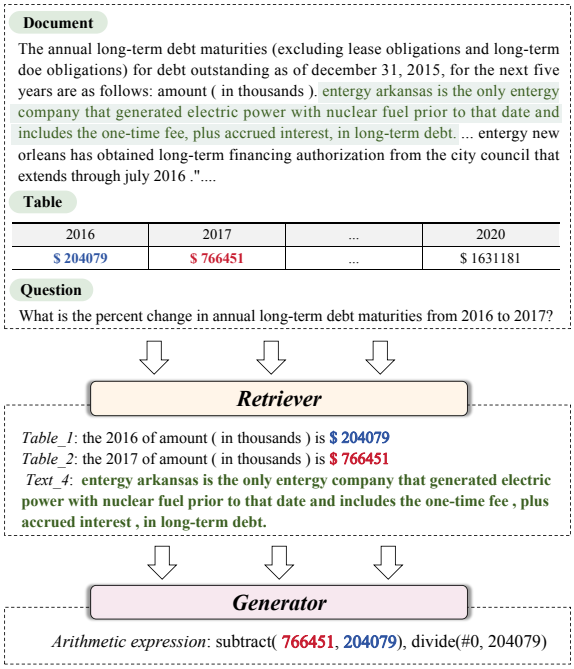

Figure 1: Retriever-generator framework for financial numerical reasoning.

The typical technology for long-form numerical reasoning is the retriever-generator framework proposed by FinQANet (Chen et al., 2021). The retriever identifies relevant information as critical facts from the long-form documents and tables, and the generator takes the critical facts as the input, generates a sequence consisting of numbers and operators. ELASTIC (Zhang and Moshfeghi, 2022) proposes an adaptive symbolic compiler and uses it to generate the expression. DyRRen (Li et al.,

2023) proposes a novel retriever-reranker-generator framework, and proposes a dynamic rerank method for location of critical facts. APOLLO (Sun et al., 2022) proposes using reinforcement learning to assist model training, enhancing model performance by improving the generated expressions.

Although there are various iterations of the retriever-generator architecture, these models encounter limitations in processing key facts, particularly due to the specificities of the financial domain and the complexities associated with tabular data, which lead to the absence of key information and thus affect the model's performance.

To solve these problems, we propose a **H**eterogeneous **G**raph **G**aussian **G**eneration model (**HG**^{3}), which uses the information augmentation to improve the performance of the retriever-generator on financial numerical reasoning tasks. For the retriever, we used heterogeneous graphs to model the relationships between documents and tables to achieve *structured information augmentation*. This approach not only allows us to capture the structural attributes of tables, but also to obtain information at different granularities, and it also can enhance the inferential capability of the retriever and find critical factual information, thereby improving retrieval effectiveness. For t he generator, we use Gaussian process stochastic functions to introduce context-aware controllable variables into the encoder, resulting in high-quality and diverse text for *controllable information augmentation*. The method can ensure that both cost and time are controllable. This random function introduces changes only in the encoder and is orthogonal to diversity-promoting decoding strategies on the decoder side. By employing different decoding strategies, various variations can be obtained, allowing the model to have more generation possibilities without losing the key fact, thereby enhancing the performance of the generator. Experimental results have indicated that the **HG**^{3} achieves a great performance on both FinQA and ConvFinQA.

The contributions of this work are as follows:

- In the retriever, we propose using heterogeneous graphs to obtain information at different granularities from documents and tables, while also incorporating the structural information of tables. This enhances the model's ability to discern details, thereby improving the effectiveness of the retriever.
- In the generator, we introduce a controllable information augmentation employing context-sensitive feature perturbation to achieve dynamic diversity scaling. This expands the model's multi-dimensional output space, effectively boosts the model's generative capabilities and lifts its overall performance.
- We propose the **H**eterogeneous **G**raph **G**aussian **G**eneration model (**HG**^{3}), which beyond all baselines, and achieves execution accuracy of 69.04($\uparrow$ 1.05%), and program accuracy of 67.97($\uparrow$ 2.37%) on test set.

2 Methodology

In this section, we describe our approach in detail. The structure of **H**eterogeneous **G**raph **G**aussian **G**eneration model (**HG**^{3}) as shown in Figure 2. Taking information augmentation as the core idea, in the retriever, we utilize Graph Attention Network (GAT) to capture the relationships between table contents and documents in the retriever to achieve *structured information augmentation* (Section 2.2.2). In the generator, Gaussian Processes are employed to enhance the diversity of representations to realize *controllable information augmentation* (Section 2.3.2).

2.1 Task Definition

Given a question q along with a numerical table T containing m rows $\{t_1, t_2, ..., t_m\}$ and a long-form document $D = \{d_1, d_2, ..., d_n\}$, the model is designed to generate arithmetic expressions that calculate the corresponding answers to a given question. However, considering the length of the document, the task is split into two subtasks. Initially, the necessary facts relevant to answering the question are retrieved from the document. Based on the retrieved facts, the generation process is subsequently executed.

2.2 Structured Information Augmentation Retriever

In the retriever, we enhance the structured information by using the row and column information of the table. The model is divided into three parts as shown below. The data encoder is utilized for encoding documents and tables, and the heterogeneous graph module is designed to model the interactions between documents and tables as well as the internal connections within tables, and the relevance scorer is used to explore facts related to the questions.

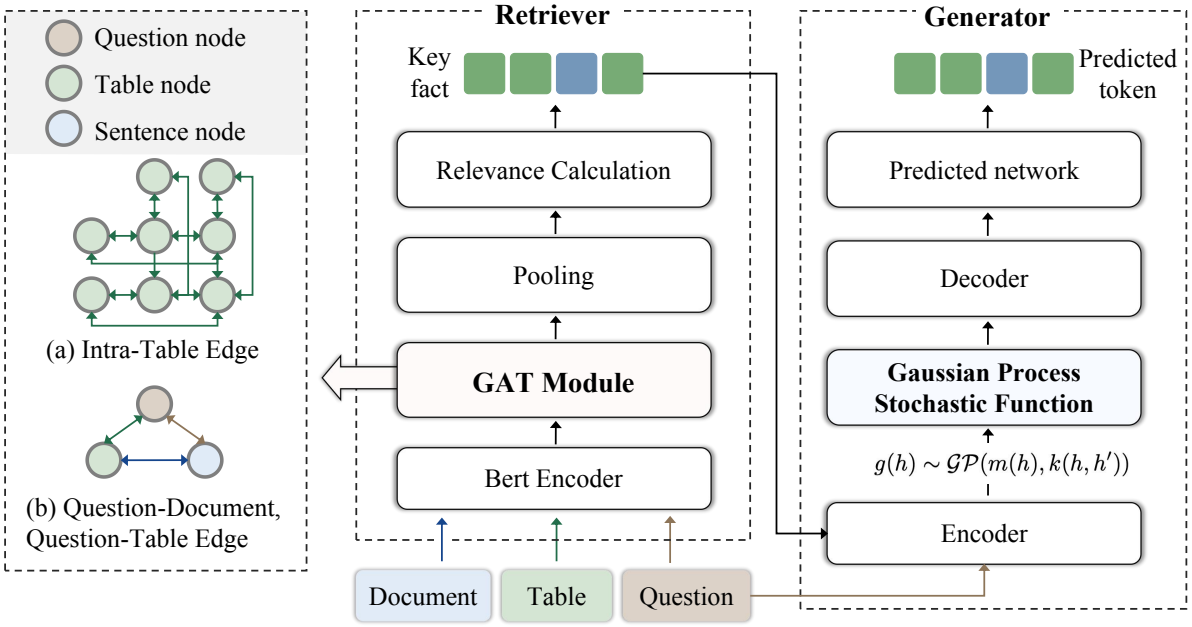

Figure 2: The **H**eterogeneous **G**raph **G**aussian **G**eneration model (**HG**^{3}).

2.2.1 Data Encoder

We first utilize a standardized template to convert table data T into text format. This process transforms each cell within the table into a well-structured sentence, resulting in a more effective and comprehensive representation of the table content (Li et al., 2023; Chen et al., 2021). The template is "the column name of row name is cell value". The sentence set D' can ultimately be expressed as follows:

$$D' = D \cup \{d_{n+1}, d_{n+2}, ..., d_{n+m}\} \quad (1)$$

where $\{d_{n+1}, d_{n+2}, ..., d_{n+m}\}$ is the sentence subset converted from rows.

Finally, the representation of documents and questions is obtained by BERT (Devlin et al., 2018):

$$\begin{aligned} \mathbf{h}^D &= \texttt{BERT}(d_1, d_2, ..., d_{m+n}) \\ \mathbf{h}^Q &= \texttt{BERT}(x_1, x_2, ..., x_{|Q|}) \end{aligned} \quad (2)$$

2.2.2 Heterogeneous Graph Construction

To model the connections between table contents and the interactions between tables and documents, a heterogeneous Graph (hG) is constructed. hG has three different kinds of nodes: table, document, and question. Each table node denotes one cell of the table data. And hG also has some document nodes that aim to model the document information. Meanwhile, hG has one question node. We argue that this node could help the model retrieve the key facts.

There are four types of edges in hG:

- **Intra-Table Edge**: The data in the same column or row is connected with intra-table edge, which facilitates the modeling of the table's structural information.
- **Intra-Document Edge**: Each sentence is connected with intra-document edges, thereby enabling the modeling of interactions between table content and the document context.
- **Question-Document Edge**: All sentences are linked to the question node with question-document edges.
- **Question-Table Edge**: Each table cell is connected to the question node with question-table edges. These connections enable the question node to aggregate information from all table cells and establish interactions between the document context and the table data.

Next, we apply GAT (Velickovic et al., 2018) on the hG to aggregate features from neighboring nodes. For a given node u, the aggregation operation can be defined as follows:

$$e_u^{(l+1)} = \sigma(\sum_{k \in K} \sum_{v \in N_k(u)} \alpha_{uv}^k W_k e_v^l + b_k) \quad (3)$$

$$e_{vu} = \text{LeakyReLU}\left(a^\top [e_v \,||\, e_u]\right) \tag{4}$$

$$\alpha_{uv}^k = \frac{exp(e_{uv})}{\sum_{w \in N_k(u)} exp(e_{uw})} \tag{5}$$

where K is the number of edge types. W_k and b_k are trainable parameters with edge type k. $N_k(u)$ denotes neighboring nodes connected to node u with edges of type k. σ is the activation function, and $a^\top$ represents the attention parameters.

We argue that different layers of GAT capture features at varying levels of abstraction. To incorporate features from all these levels, we concatenate the hidden states from each layer to form the representation of node u, and a pooling layer is used to obtain the final representation:

$$\mathbf{h}_u = \texttt{Mean}([e_u^0, e_u^1, e_u^2, ..., e_u^N]) \tag{6}$$

where e_u^0 is the initial representation of node u, and N is the number of layers. And we use the mean representation of the entire sentence to initialize the node representation.

2.2.3 Relevance Scorer

After obtaining the final representation, we sort each text paragraph and table column with the correlation confidence of the model output:

$$\texttt{Sim}(\mathbf{q}, \mathbf{d}_i) = \texttt{Cos}(\mathbf{q}, \mathbf{h}_i) \tag{7}$$

where $d_i \in D$ and $h_i \in h^D$.

The top-k evidences is selected as the retrieval results. Subsequently, these evidences are concatenated with the question to form the input for the generator model.

2.3 Controllable Information Augmentation Generator

In the generator, we achieve the controllable information augmentation by Gaussian process. We use a seq2seq model as the generator, which comprises a pre-trained encoder and a decoder. We propose a novel module for learning rich contextual representations by transforming the deterministic hidden states from the encoder into stochastic hidden states using stochastic functions.

As mentioned above, the question and top-k retrieved facts are concatenated and fed into a BERT encoder to produce the representation h^q and h^D.

2.3.1 Stochastic Function

To mitigate information interference during reasoning and introduce context-aware variability into the encoder, we propose a stochastic mapping function $g()$ that converts deterministic hidden states into random context variables:

$$p(z|h) = g(h) + \epsilon \tag{8}$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ is a Gaussian noise. Then, the decoder input z is obtained, which is sampled from $p(z|h)$. This allows the model to get more inference information from contextual dependencies.

2.3.2 Gaussian process prior

The stochastic mapping function $g()$ is pivotal for the model's efficacy. It introduces variability into the hidden states while retaining their informational integrity. This mechanism facilitates the introduction of controllable perturbations, thereby enhancing inference diversity while maintaining computational efficiency in both train and inference phases. As shown in Figure 3, through this controllable information Augmentation, we have increased the possibilities of the final expression.

We propose constructing the stochastic mapping function $g()$ by incorporating a Gaussian process (GP) as its functional prior. With the GP prior, we can sample multiple stochastic functions to generate the final sequence. The stochastic function $g()$ is defined according to the GP prior:

$$g(h) \sim GP(m(h), k(h, h')) \tag{9}$$

$$m(h) = h$$

$$k(h, h') = \beta^2 exp - \frac{\|h - h'\|_2^2}{2\gamma^2} \tag{10}$$

where h is the hidden state, h' indicates the other contextual hidden states. The parameter β controls the variation intensity between the sampled function $g(h)$ and the mean function $m(h)$. The parameter γ controls the covariance among the random variables. Increasing γ will reduce the distinction between the sampled values.

To ensure that the sampled random states z retain all the information from h, We employ a semi-parametric GP prior (Murphy, 2012). This approach endows our model with more diverse inputs.

Then, the decoder takes the embedding z as input and decodes the numerical reasoning program step by step:

$$p(y_t|y_{t-1}, z) = Decoder(y_{t-1}, z) \tag{11}$$

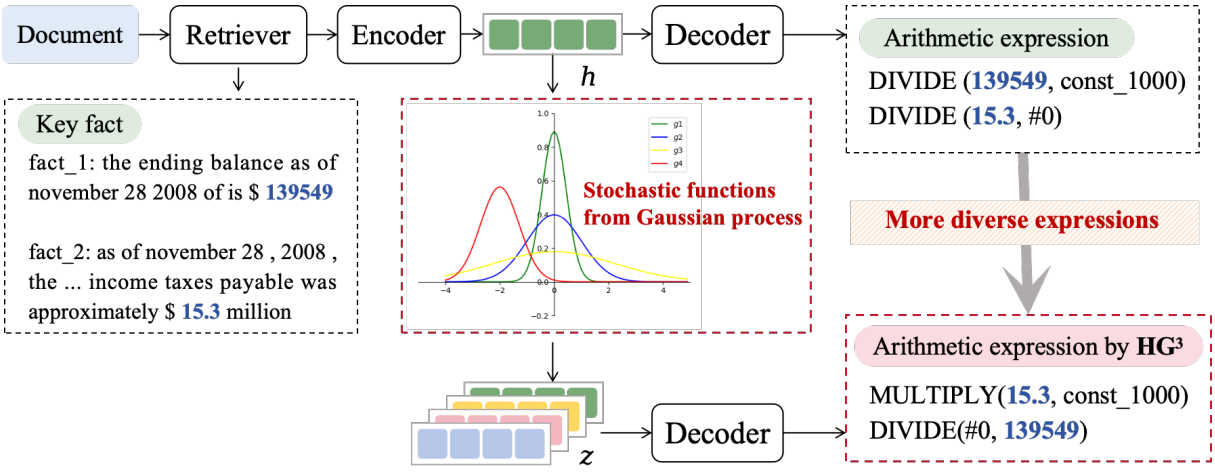

Figure 3: Comparison between HG[3] and other models.

where y_t is the generated token in step t.

2.3.3 Loss Function

To simplify the inference procedure, we use variational inference to approximate the GP posterior $p(g|h)$ and use maximum likelihood estimation to learn other parameters. Specifically, we approximate the true posterior $p(z|h, y)$ with the variational posterior $q(z|h, y)$ by maximizing the evidence lower bound of the marginal log-likelihood (ELBo):

$$\begin{aligned}\log p(y|h) \geq & \mathbf{E}_q[\log p(y|z)] \\ & - \mathrm{KL}[q(z|h, y)\|p(z|h)]\end{aligned} \tag{12}$$

where $p(z|h)$ is obtained by Equation 8. And to further simplify the approximation of $q(z|h, y)$, we use the mean-field amortized variational approximation(Kingma and Welling, 2014) to approximate the parameters:

$$\begin{aligned}q(z|h) &\approx \prod_{i=1}^{N} q(z_i|h_i) \\ &= \prod_{i=1}^{N} \mathcal{N}(f_\mu(h_i), f_{\sigma^2}(h_i))\end{aligned} \tag{13}$$

where $f_\mu()$ and $f_{\sigma^2}()$ represent the mean and covariance in the amortized variational inference network, and z only rely on hidden states h.

3 Experiments

In this section, we provide a detailed overview of the datasets employed for the evaluation of the reasoning task.

3.1 Datasets

We evaluate our model on FinQA (Chen et al., 2021) and ConvFinQA (Chen et al., 2022).

FinQA is a numerical reasoning dataset comprising 8,281 examples with fully annotated reasoning programs, derived from the publicly available earnings reports of S&P 500 companies over a decade (Zheng et al., 2021). The data is divided into train (6,251), dev (883), and test (1147) sets according to a split of 75%/10%/15%. Each question includes a table and a long-form document, with an average token count of 687.53 and a maximum of 2,679 tokens. Notably, 53.70% of the examples contain two or more factual pieces, and 40.90% of the reasoning programs involve multiple steps.

ConvFinQA is a conversational numerical reasoning dataset comprising 3,892 dialogues, which include a total of 14,115 questions. And the dataset is split into 3,037/421/434 for train/dev/test sets.

3.2 Metrics

3.2.1 Retriever

In the retriever, we evaluate our model using Recall@3 and Recall@5, which evaluate the model by calculating the percentage of correctly identified positives among all positive predictions. Since each sample may have multiple positive predictions, we assume that the top N predictions in Recall@N are all positive.

3.2.2 Generator

In the generator, We evaluate our model with Program Accuracy (Prog Acc) and Execution Accuracy (Exe Acc). Prog Acc assesses the syntactic equivalence between generated arithmetic expres-

Model	Dev		Test	
	Exe Acc	Prog Acc	Exe Acc	Prog Acc
Longformer (Beltagy et al., 2020)	23.83	22.56	21.90	20.48
NeRd (Ran et al., 2019)	47.53	45.37	48.57	46.76
ELASTIC (Zhang and Moshfeghi, 2022)	65.00	61.00	62.16	57.54
BERT				
FinQANet (Chen et al., 2021)	49.91	47.15	50.00	48.00
DyRRen (Li et al., 2023)	61.16	58.32	59.37	57.54
HG^3 (ours)	**65.62**	**64.13**	**64.92**	**63.96**
RoBERTa				
FinQANet (Chen et al., 2021)	61.22	58.05	61.24	58.86
DyRRen (Li et al., 2023)	66.82	63.87	63.30	61.29
APOLLP (Sun et al., 2022)	69.70	65.91	67.99	65.60
HG^3 (ours)	**70.96**	**69.21**	**69.04**	**67.97**
Human Expert	-	-	91.16	87.49
General Crowd	-	-	50.68	48.17

Table 1: Comparison of HG^3 and baselines on FinQA. The pre-trained models that are used in the experiments are BERT-base-uncased and RoBERTa-large.

Model	Dev		Test	
	Exe Acc	Prog Acc	Exe Acc	Prog Acc
T-5	58.38	56.71	58.66	57.05
GPT-2	59.12	57.52	58.19	57.00
RoBERTa				
FinQANet	68.32	67.87	68.90	68.24
APOLLO	**76.47**	74.14	76.00	74.56
HG^3 (ours)	76.18	**74.83**	**77.86**	**76.24**
Human Expert	-	-	89.44	86.34
General Crowd	-	-	46.90	45.52

Table 2: Comparison of HG^3 and baselines on ConvFinQA.

sions and the golden expressions, while Exe Acc evaluates whether the generated expressions produce the correct results. Considering multiple valid solutions, Exe Acc is essential to ensure functional correctness beyond syntactic similarity.

3.3 Baselines

We compare our model to other publicly available methods, including: (1) **ELASTIC** (Zhang and Moshfeghi, 2022), which utilizes an adaptive symbolic compiler to get the expression. (2) **NeRd** (Ran et al., 2019), which proposes a novel expression generator based on a pointer network. (3) **FinQANet** (Chen et al., 2021), which first proposes a retriever-generator framework designed to generate arithmetic expressions from both tabular and textual data. (4) **Longformer** (Beltagy et al., 2020), which processes entire long documents to generate arithmetic expressions. (5) **DyRRen** (Li et al., 2023), which extends the retriever-generator framework by incorporating dynamic reranking of retrieved facts to enhance reasoning capabilities. (6) **APOLLO** (Sun et al., 2022), which uses reinforcement learning to normalize the generated expressions. (7) **Human performance**, which includes both experts and non-experts participants sourced from the original paper(Chen et al., 2021).

3.4 Implementation Details

Our model is implemented using PyTorch and the Transformer architecture (Vaswani et al., 2017) from Huggingface, and evaluated on a single NVIDIA V100 32GB GPU. In the retriever, we employ the pre-trained language models BERT-base-uncased and RoBERTa-large to obtain token representations. The hyperparameter k is set to 3 to retrieve the Top-3 ranked facts as the output. Meanwhile, the number of epochs is set to 30, the batch size is set to 4, and the maximum sequence length is set to 256. We optimize the retriever using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of $2e$-5 to update the model parameters. In the generator, the number of epochs is set to 500, the batch size is set to 8 for the encoder and decoder. We use Adam with a learning

Model	FinQA(Dev)		FinQA(Test)		ConvFinQA(Dev)	
	R@3	R@5	R@3	R@5	R@3	R@5
BERT						
FinQANet	88.35	90.96	87.24	90.65	86.43	89.74
DyRRen	89.74	91.38	88.12	90.14	87.32	90.03
APOLLP	90.26	91.32	88.94	90.45	89.21	91.71
Ours	91.40	92.28	89.41	91.18	90.42	92.93
RoBERTa						
FinQANet	91.30	93.89	89.82	93.22	88.95	92.74
DyRRen	92.36	94.61	90.34	93.41	89.72	93.16
APOLLP	93.58	95.62	91.76	93.95	91.67	94.56
Ours	94.79	96.47	92.89	94.98	92.74	95.83

Table 3: The experimental results of retriever Recall Top-3 and Top-5 in FinQA and ConvFinQA. The test set on ConvFinQA does not have ground truth for retrieved facts.

Model	Dev		Test	
	Exe Acc	Prog Acc	Exe Acc	Prog Acc
BERT				
HG^3	**65.62**	**64.13**	**64.92**	**63.96**
w/o Graph	64.34	63.26	63.48	62.20
w/o GPSF	63.94	62.32	62.51	61.24
RoBERTa				
HG^3	**70.96**	**69.21**	**69.04**	**69.97**
w/o Graph	69.39	67.59	68.12	69.01
w/o GPSF	68.79	67.06	66.72	67.43

Table 4: Ablation study on FinQA.

rate of $2e$-5 for BERT-base-uncased and $1e$-5 for RoBERTa-large. The maximum sequence length is set to 256. We clip the gradients of model parameters to a max norm of 1.0. Additionally, we adopt a linear warm-up (Targ et al., 2016) for the first 10% of steps followed by a linear decay to 0 to prevent gradient explosion and over-fitting.

3.5 Main Result

Table 1 presents the performance of HG^3 and baselines on FinQA. HG^3 achieves the best performance, with Exe Acc and Prog Acc scores of 70.96 and 69.21 on the dev set, and 69.04 and 67.97 on the test set, respectively. The results demonstrate that models utilizing RoBERTa significantly outperform those using BERT. This highlights the importance of incorporating sufficient relevant knowledge to enhance model reasoning capabilities.

Table 2 shows the results on ConvFinQA. Compared with FinQANet, HG^3 exceeds 0.69% of Prog Acc on the dev set, 1.86% of Exe Acc and 1.68% of Prog Acc on the test set. These results indicate that our model exhibits strong performance and robustness in multi-turn conversational numerical reasoning tasks.

HG^3 outperforms APOLLO both on BERT and RoBERTa, surpassing 1.26% of Exe Acc and 3.30% of Prog Acc on the test set, 1.05% of Exe Acc and 2.37% of Prog Acc on the dev set. This indicates that improving the inferential capability of the retrieval-generator framework is necessary for financial numerical inference tasks.

And it can be noted that HG^3, whether using BERT or RoBERTa, has surpassed the level of the general crowd, but there is still a gap compared to the human expert.

Table 3 reports the retriever performance. Our model achieves superior results on both FinQA and ConvFinQA datasets, outperforming APOLLO across BERT and RoBERTa implementations. Specifically, on the FinQA test set, our model surpasses APOLLO by 1.13% in Top-3 and 1.03% in Top-5; on the FinQA dev set, it achieves gains of 1.21% in Top-3 and 0.85% in Top-5 for the RoBERTa version. It demonstrates that using graphs to construct table attributes mitigates the impact of templates on table structure, thereby enhancing retrieval effectiveness.

3.6 Ablation Study

Our model comprises two essential components: the Holistic Graph module and the Gaussian Process Stochastic Function module. The variant w/o Graph replaces our retriever with the original retriever. w/o GPSF removes the Gaussian Process Stochastic Function module, directly generating arithmetic expressions based on retrieval facts.

As shown in Table 4, w/o Graph in BERT leads to a relative drop of Exe Acc and Prog Acc by 1.44%, 1.76% on the test set. The performance of w/o Graph in RoBERTa exhibits a 0.92% reduction in Exe Acc and a 0.96% reduction in Prog Acc compared to ours. These results underscore the necessity of using graphs to capture relationships between tables, as the structural attributes of tables facilitate the retrieval of key facts. Moreover, the interaction between documents, questions, and table contents is crucial to filter the final key facts.

Ignoring the Gaussian Process Stochastic Function module leads to a significant drop in performance. In BERT, w/o GPSF performs 2.41% lower in Exe Acc and 2.72% lower in Prog Acc than ours in BERT version on the test set. Similarly, in RoBERTa version, w/o GPSF results

Model	Dev		Test	
	Exe Acc	Prog Acc	Exe Acc	Prog Acc
ours (BERT)	**65.62**	**64.13**	**64.92**	**63.96**
Type				
table-only	75.32	73.84	73.63	71.68
sentence-only	51.06	49.47	55.19	54.31
table-sentence	43.76	40.62	39.84	37.62
Expression Number				
1	70.21	68.34	69.48	68.17
2	65.51	62.40	63.18	60.04
> 2	29.44	27.24	33.43	31.99
DyRRen (BERT)	61.16	58.32	59.37	57.54
Type				
table-only	72.51	69.37	68.98	66.71
sentence-only	46.46	44.44	49.47	48.76
table-sentence	38.46	35.66	34.18	32.28
Expression Number				
1	66.16	63.67	64.37	63.00
2	60.98	57.49	57.46	54.77
> 2	26.03	23.29	29.76	28.57

Table 5: Fine-grained comparison with our model, DyRRen on by question type and expression number.

in a performance decrease of 2.32% in Exe Acc and 2.54% in Prog Acc compared to ours. This highlights the effectiveness of Gaussian Process Stochastic Function module. This further indicates that introducing controlled perturbations to latent variables enhances the diversity of the generated results, thereby improving overall performance.

3.7 Fine-Grained Results

As shown in Table 5, the HG^3 achieves the best performance in all fine-grained analysis on dev set and test set. Regarding question types, HG^3 outperforms the baselines in all types. For table-only questions, HG^3 performs 4.65% higher in Exe Acc and 4.97% higher in Prog Acc than DyRRen in BERT version on test set. For sentence-only questions, HG^3 in BERT leads to a relative drop of Exe Acc and Prog Acc by 5.72%, 5.55% on the test set. For table-sentence questions, HG^3 performs 5.66% higher in Exe Acc and 5.34% higher in Prog Acc than DyRRen in BERT version on test set. It can be seen that our model performs exceptionally well on sentence-only and table-only questions, indicating that it effectively addresses problems of individual types. However, a gap remains in handling table-sentence types, indicating that mixed table and document problems remain challenging.

For expression number, HG^3 surpasses other models in all cases. For Expression Number is 1, HG^3 performs 5.11% higher in Exe Acc and 5.17% higher in Prog Acc than DyRRen in BERT version on test set. For Expression Number is 2, HG^3 in BERT leads to a relative drop of Exe Acc and Prog Acc by 5.71%, 5.27% on the test set. For Expression Number more than 2, HG^3 performs 3.67% higher in Exe Acc and 3.42% higher in Prog Acc than DyRRen in BERT version on test set. It can be seen that for problems with fewer than two expressions, our model already performs well. However, it can also be seen that our model's improvement in this type is not as significant as in other types. This is due to the limitation that the facts input to the generator are fixed. It demonstrates that only by providing the model with sufficient context can accurate results be obtained. When the number of expressions exceeds 2, although our model's performance is not as good as for those with fewer than two expressions, it still shows significant improvement compared to other methods. This indicates that our method is effective, and introducing controlled perturbations to latent variables helps in generating longer expressions, leading to performance enhancement.

4 Conclusion

We propose HG^3, comprising two innovative modules to enhance the performance of the model: 1) The heterogeneous graph module models the structural attributes of tables, enhancing the interaction between documents and tables, thereby improving retrieval effectiveness. 2) The Gaussian process stochastic function module provides more diversity in the generation process, enhancing the model's representation capability and thereby improving the generation effectiveness. Our model outperforms all baselines on FinQA and ConvFinQA.

Limitations

Our model has some limitations. The number of key facts passed to the generator is fixed, which significantly restricts the model's capabilities. For problems with more facts, this means that even humans cannot solve such issues. Additionally, although we have introduced diversity to increase the model's upper limit, overly diverse generated results can lead to errors, such as incorrect operators, which also affect the model's performance. In the future, to address these issues, we will expand our model to increase its capacity for handling varying numbers of facts and build an adaptive retriever. We will also impose restrictions on the generator's output to ensure better model performance.

References

Gerardo Adesso. 2022. Gpt4: The ultimate brain. *Authorea Preprints*.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. 2020. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. *arXiv preprint arXiv:2004.07347*.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, et al. 2021. Finqa: A dataset of numerical reasoning over financial data. *arXiv preprint arXiv:2109.00122*.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. 2022. Convfinqa: Exploring the chain of numerical reasoning in conversational finance question answering. *arXiv preprint arXiv:2210.03849*.

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

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 887–896.

Zhanming Jie, Jierui Li, and Wei Lu. 2022. Learning to reason deductively: Math word problem solving as complex relation extraction. *arXiv preprint arXiv:2203.10316*.

Ziqi Jin and Wei Lu. 2023. Tab-cot: Zero-shot tabular chain of thought. *arXiv preprint arXiv:2305.17812*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *stat*, 1050:1.

Xiao Li, Yin Zhu, Sichen Liu, Jiangzhou Ju, Yuzhong Qu, and Gong Cheng. 2023. Dyrren: A dynamic retriever-reranker-generator model for numerical reasoning over tabular and textual data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13139–13147.

Kevin P Murphy. 2012. *Machine learning: a probabilistic perspective*. MIT press.

Rungsiman Nararatwong, Natthawut Kertkeidkachorn, and Ryutaro Ichise. 2022a. Enhancing financial table and text question answering with tabular graph and numerical reasoning. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*, pages 991–1000.

Rungsiman Nararatwong, Natthawut Kertkeidkachorn, and Ryutaro Ichise. 2022b. Kiqa: Knowledge-infused question answering model for financial table-text data. In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 53–61.

Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. Numnet: Machine reading comprehension with numerical reasoning. *arXiv preprint arXiv:1910.06701*.

Jiashuo Sun, Hang Zhang, Chen Lin, Yeyun Gong, Jian Guo, and Nan Duan. 2022. Apollo: An optimized training approach for long-form numerical reasoning. *arXiv preprint arXiv:2212.07249*.

Sasha Targ, Diogo Almeida, and Kevin Lyman. 2016. Resnet in resnet: Generalizing residual architectures. *arXiv preprint arXiv:1603.08029*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. *stat*, 1050:4.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Qinzhuo 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.

Jiaxin Zhang and Yashar Moshfeghi. 2022. Elastic: numerical reasoning with adaptive symbolic compiler. *Advances in Neural Information Processing Systems*, 35:12647–12661.

Renhui Zhang, Youwei Zhang, and Yao Yu. 2022. A robustly optimized long text to math models for numerical reasoning on finqa. *arXiv preprint arXiv:2207.06490*.

Wei Zhao, Mingyue Shang, Yang Liu, Liang Wang, and Jingming Liu. 2020. Ape210k: A large-scale and template-rich dataset of math word problems. *arXiv preprint arXiv:2009.11506*.

Xinyi Zheng, Douglas Burdick, Lucian Popa, Xu Zhong, and Nancy Xin Ru Wang. 2021. Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 697–706.

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. *arXiv preprint arXiv:2105.07624*.

A Related Works

Math Word Problem (MWP) (Huang et al., 2016) is a challenging task in numerical reasoning. The goal of this task is to generate and calculate arithmetic expressions to answer questions. Based on MWP question descriptions, researchers have proposed several rule-based methods, such as statistical learning (Hosseini et al., 2014), graph-based techniques (Wu et al., 2021), and tree-based methods (Jie et al., 2022). However, as the understanding of these problems has deepened, the descriptions have become longer and more complex, reducing the effectiveness of these methods. The MathQA (Amini et al., 2019) and Ape210k (Zhao et al., 2020) datasets demonstrate that MWP questions consist only of short texts without tables or additional sentences, distinguishing them from other tasks. Additionally, The HybridQA (Chen et al., 2020) introduces a new task by combining texts and tables.

Researchers have since shifted their focus to long-form numerical reasoning tasks involving tabular data, which present greater challenges than traditional MWP tasks. Datasets like TAT-QA (Zhu et al., 2021), FinQA (Chen et al., 2021), and ConvFinQA (Chen et al., 2022) are examples of such hybrid datasets derived from financial reports. Unlike MWP, these tasks require retrieving facts from extensive documents and tables and then calculating the answers. (Zhang et al., 2022) propose a method that leverages the unique strengths of various specialized models, combining them to achieve enhanced performance. Nararatwong et al.(Nararatwong et al., 2022b) employs a knowledge injection method to address the issue of hard-to-understand operators and utilizes a GNN to resolve the problem of table data structure disruption. (Nararatwong et al., 2022a).

These methods overlook the structural attributes of tables, with some even failing to enhance reasoning capabilities through retrieval methods, thereby not adequately addressing the issue. Additionally, they do not sufficiently focus on the generation process, which limits their generative capabilities. To address these limitations, we incorporate the structural attributes of tables to compensate for the missing information during retrieval. Simultaneously, we introduce controllable perturbations into the generation process through a Gaussian process, enabling the generator to produce higher-quality results.

With the rise of large language models, their powerful few-shot reasoning capabilities without fine-tuning have gained increasing attention. Wei et al.(Wei et al., 2022) proposes the chain-of-thought approach, enabling LLMs to generate their own reasoning processes, which has garnered significant attention. Further studies have found that large language models (LLMs) exhibit varying performance when generating different types of reasoning processes(Jin and Lu, 2023). In addition to LLMs, researchers have discovered that fine-tuning small-scale models using reasoning processes generated by LLMs can also improve performance(Ho et al., 2022). Given the lower computational overhead and acceptable performance of small-scale models, this area remains a valuable topic for research.

Despite their impressive performance in numerical reasoning tasks, the substantial computational overhead of LLMs limits their practical application. Therefore, we aim to achieve the performance of LLMs on these tasks using small-scale models. By increasing the diversity of the generation, we hope to enhance the robustness of the models.