DyRRen: A Dynamic Retriever-Reranker-Generator Model for Numerical Reasoning over Tabular and Textual Data

Xiao Li, Yin Zhu, Sichen Liu, Jiangzhou Ju, Yuzhong Qu, Gong Cheng*

State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
{xiaoli.nju, yinzhu, sichenliu, jujiangzhou}@smail.nju.edu.cn, {yzqu,gcheng}@nju.edu.cn

Abstract

Numerical reasoning over hybrid data containing tables and long texts has recently received research attention from the AI community. To generate an executable reasoning program consisting of math and table operations to answer a question, state-of-the-art methods use a retriever-generator pipeline. However, their retrieval results are static, while different generation steps may rely on different sentences. To attend to the retrieved information that is relevant to each generation step, in this paper, we propose DyRRen, an extended retriever-reranker-generator framework where each generation step is enhanced by a dynamic reranking of retrieved sentences. It outperforms existing baselines on the FinQA dataset.

1 Introduction

Numerical reasoning has drawn much attention in machine reading comprehension tasks. Previous datasets on numerical reasoning like DROP (Dua et al. 2019) mainly focus on simple calculations and comparisons. Recently in this field, more complicated arithmetic expression generation takes an important part, especially in the context of financial reports. Recognized datasets like FinQA (Chen et al. 2021b) and TAT-QA (Zhu et al. 2021) involve hybrid question answering (QA), i.e., QA over information represented in heterogenous forms including tables and texts.

To acquire an answer by numerical reasoning, a standard method is to generate an arithmetic expression from structured tables and unstructured sentences. For instance, Figure 1 shows an example in FinQA. The context includes a table describing sales records in different years, and several sentences stating relevant financial information. According to the question, one needs to determine arithmetic operators `MINUS` and `DIVIDE`, and select their arguments from either tables or sentences to generate the correct arithmetic expression and calculate the answer. In some cases, an argument of an arithmetic expression (e.g., `#0` in Figure 1) could be the result of some former expression.

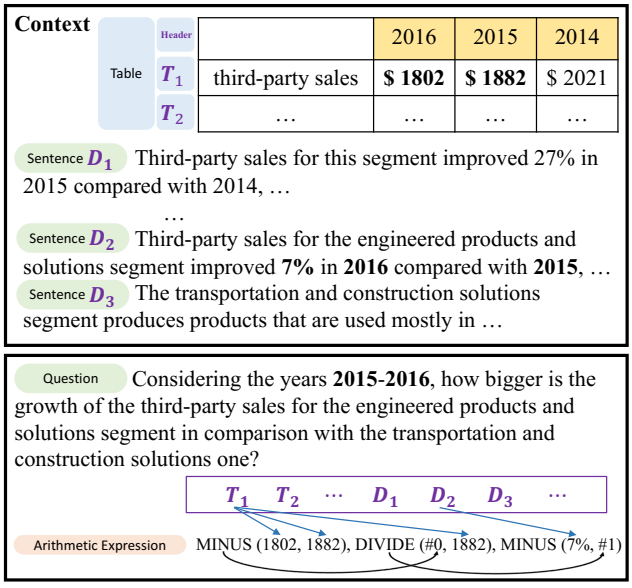

Figure 1: A question sampled from the FinQA dataset.

1.1 Existing Methods and Limitations

Intuitively, the generation of an arithmetic expression can be divided into two modules: Retriever and Generator. The retriever filters the most relevant information as the input to the generator which then predicts expressions using operators predefined in a domain specific language (DSL) and numbers extracted from the input. FinQANet (Chen et al. 2021b) is one such state-of-the-art method. Its retriever converts a table into sentences. Then a BERT (Devlin et al. 2019) binary classifier is applied to identify relevant sentences, based on which an encoder-decoder model is implemented to generate an arithmetic expression. Encoders can be BERT or RoBERTa (Liu et al. 2019) which are both encoder-only models. The hidden layer output of an LSTM (Hochreiter and Schmidhuber 1997) is used to predict expression tokens.

We observe the following limitations of FinQANet.

First, in FinQANet's generator, retrieved sentences are concatenated into a long sequence and inputted into the encoder, from which different numbers are extracted at different generation steps. Existing models have difficulty in locating the correct number from a long sequence. For example, when generating the argument "7%" in Figure 1, the generator should pay exclusive attention to the unique

*Corresponding author
Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

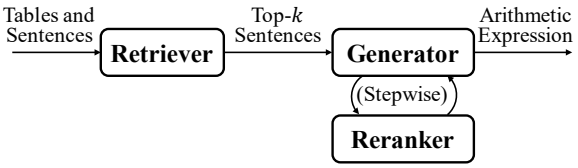

Figure 2: Overview of DyRRen.

retrieved sentence D_2 which contains "7%". Therefore, it would be helpful to design a mechanism that can dynamically adjust the attention of the generator at generation time.

Second, there is plenty of room to improve FinQANet's retriever which is a simple binary classifier. Indeed, it could not fully exploit the relationships between question and context. For example, in the question in Figure 1, the retriever should focus on particular entities such as "2015", "2016", and "third-party sales", which requires intensive interaction at the token level. Moreover, it could not directly capture the differences between positive and negative sentences.

1.2 Our Approach

We overcome the above limitations with our novel framework DyRRen, short for **Dy**namic **R**etriever-**R**eranker-g**en**erator. Figure 2 sketches our framework. DyRRen switches its attention to different sentences at different generation steps via a novel dynamic reranking mechanism.

Generator In the generator, we focus on scoring the input tokens and looking for the next token most likely to appear in the generated expression. This module is implemented by an LSTM with the attention mechanism. To improve the accuracy of the generator, we incorporate a reranker to dynamically pick relevant sentences.

Reranker At each step of generation, we score and rerank retrieved sentences in order to find one that contains the target number. Specifically, we take advantage of the relationships among the decoder output of the generator, formerly generated expression, and retrieved sentences to dynamically rerank sentences at each step to enhance the generator.

Retriever Dense retrieval is used to replace the simple BERT binary classifier. To improve retrieval performance, we implement meticulous interaction between each question-sentence pair and employ a set of useful mechanisms such as token-level late interaction. The loss function is also tailored to the scenario in which the retrieval space contains multiple positive and negative sentences.

To summarize, our contributions include

- A novel retriever-reranker-generator framework for solving numerical reasoning over tabular and textual data, especially using a reranker which can dynamically locate target information at different generation steps, and
- A thoughtfully designed retriever in which we strengthen token-level and pairwise interaction.

1.3 Code

Our code is available on GitHub: https://github.com/nju-websoft/DyRRen.

1.4 Outline

We elaborate our approach in Section 2, present experiments in Section 3, discuss related work in Section 4 and conclude our work in Section 5.

2 Approach

In the task of numerical reasoning over tabular and textual data, a question Q is given together with a structured table T which consists of r rows $\{T_1, T_2, \cdots, T_r\}$ and a set of n unstructured sentences $\mathbb{D} = \{D_1, D_2, \cdots, D_n\}$. The goal is to generate an arithmetic expression $G = (\text{op}_1(\text{args}_{11}, \text{args}_{12}), \text{op}_2(\text{args}_{21}, \text{args}_{22}), ...)$, where op_i is a binary operator such as `DIVIDE` defined in a domain specific language (DSL) and args_{ij} is either a constant token, a memory token indicating the result of some previous operation, or a span[1] that appears in Q, T, or $\mathbb{D}$.

We extend the existing retriever-generator framework to a retriever-ranker-generator framework. We first use the retriever to retrieve the most relevant sentences to the question (Section 2.1), and then use the retrieved sentences to generate arithmetic expressions (Section 2.2) which is enhanced by a dynamic reranker of sentences (Section 2.3).

2.1 Retriever

Given the question Q, table T and sentences $\mathbb{D}$, our retriever aims to retrieve supporting facts from T and $\mathbb{D}$ to answer Q. The retriever is shown bottom left in Figure 3.

Encoder In the retriever, we process tabular data in the same way as FinQANet (Chen et al. 2021b), converting each row of the tabular data into a sentence using templates.Specifically, we convert each cell into "the **row name** of **column name** is **cell value** ;" and then we concatenate the converted cell sentences of a row into a row sentence. For example, the second row of the table in Figure 1 will be converted to "the **third-party sales** of **2016** is **$ 1802** ; the **third-party sales** of **2015** is **$ 1882** ; ...".

Recall that the retriever of FinQANet uses BERT encoder to obtain the representations of sentences (including sentences converted from tables). Then it uses a binary classifier to determine whether each sentence is relevant. Different from FinQANet's retriever, we use dense retrieval to improve the performance of the retriever.

Specifically, by using templates to convert each row of T into a sentence, we obtain a set of sentences

$$\mathbb{D}' = \{D_1, D_2, \cdots, D_n\} \cup \{D_{n+1}, D_{n+2}, \cdots, D_{n+r}\} \quad (1)$$

where $\{D_{n+1}, D_{n+2}, \cdots, D_{n+r}\}$ are converted from rows.

We use BERT to encode question Q and each sentence $D \in \mathbb{D}'$ into $\mathbf{H}^Q$ and $\mathbf{H}^D$ respectively. More details about our encoder will be introduced in Section 2.2. Inspired by ColBERT (Khattab and Zaharia 2020), we also make the following optimizations to the encoder.

- Prepend the special token `[Q]` and `[D]` to the question tokens and sentence tokens respectively. This makes it easier for BERT encoder to identify the type of text.

[1] It could be a number or a row header. For clarity, we will refer to it as a numeric argument/token in the following.

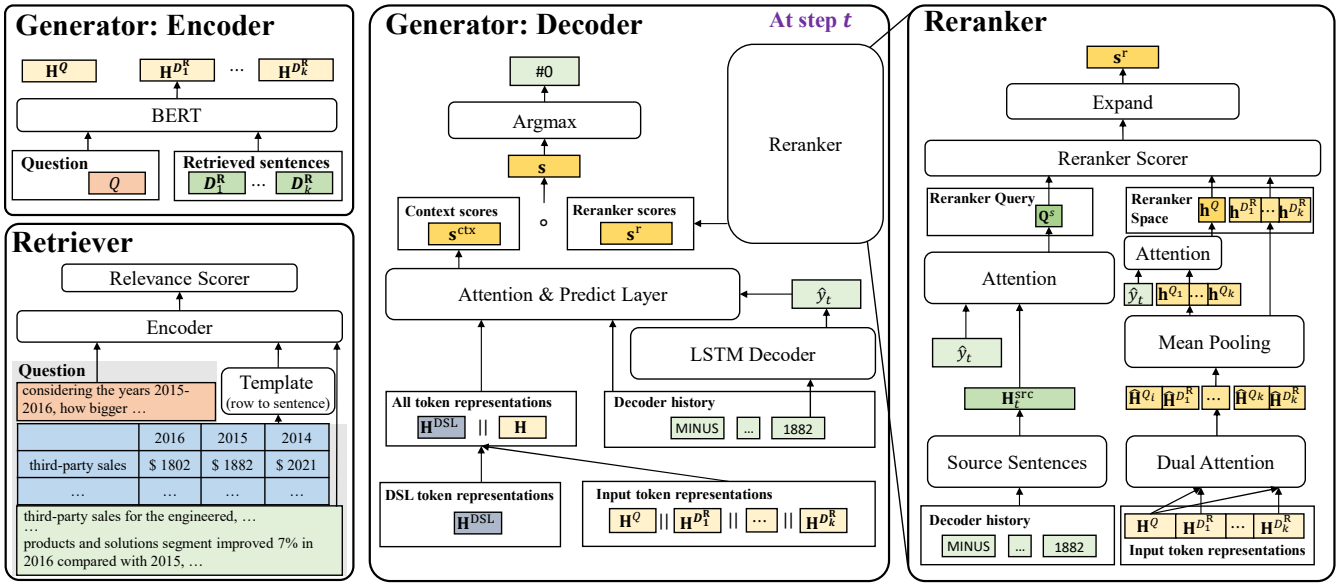

Figure 3: Implementation of DyRRen.

- We pad question tokens with BERT's special `[mask]` token up to a fixed length. The token `[mask]` was used to mask the words to be predicted while pre-training. It allows BERT encoder to generate representations that match the given question's semantics by filling `[mask]` tokens to optimize the question representation.
- Concatenate the question tokens after sentence tokens. It allows the question and sentences to interact with each other inside the BERT encoder.

Relevance Scorer Using $\mathbf{H}^Q$ and $\mathbf{H}^D$, the retriever computes the relevance score of D to Q through a late interaction mechanism to allow token-level interaction by calculating token similarity as in ColBERT:

$$\texttt{sim}(\mathbf{H}^Q, \mathbf{H}^D) = \sum_{\mathbf{H}_i^Q \in \mathbf{H}^Q} \max_{\mathbf{H}_j^D \in \mathbf{H}^D} \frac{\mathbf{H}_i^Q \cdot \mathbf{H}_j^{D^T}}{\|\mathbf{H}_i^Q\| \|\mathbf{H}_j^{D^T}\|} \quad (2)$$

where $\|\cdot\|$ represents vector modulus operator. We take top-k sentences with the highest scores as the retrieval results.

Our retriever adopts RankNet pairwise loss (Burges 2010; Zhan et al. 2021). Given the question Q, let $\mathbb{D}^+$ and $\mathbb{D}^-$ be the sets of all positive sentences and all negative sentences respectively. Our loss function is formulated as follows:

$$L(Q, \mathbb{D}^+, \mathbb{D}^-) = \sum_{D_i^+ \in \mathbb{D}^+} \sum_{D_j^- \in \mathbb{D}^-} \log(1 + \frac{\exp(\texttt{sim}(\mathbf{H}^Q, \mathbf{H}^{D_j^-}))}{\exp(\texttt{sim}(\mathbf{H}^Q, \mathbf{H}^{D_i^+}))}). \quad (3)$$

With this loss function, we compare the scores of all positives and all negatives pairwisely to allow pairwise interaction between sentences. It aims to rank all positives ahead of all negatives in the retrieval results.

2.2 Generator

Our generator extends FinQANet's generator by incorporating a dynamic reranker.

Given the question Q and the retrieved sentences $\mathbb{D}^{\mathrm{R}} = \{D_1^{\mathrm{R}}, \cdots, D_k^{\mathrm{R}}\} \subseteq \mathbb{D}'$, our generator aims to generate the arithmetic expression that the question implies. Figure 3 gives an overview of our generator (top left and center).

Encoder For question $Q = (u_1^Q, u_2^Q, \cdots, u_{|Q|}^Q)$, we directly feed it into BERT to obtain its representation:

$$\begin{aligned} \mathbf{H}^Q &= [\mathbf{h}_1^Q; \mathbf{h}_2^Q; \cdots; \mathbf{h}_{|Q|+2}^Q] \\ &= \texttt{BERT}(\texttt{[CLS]}, u_1^Q, u_2^Q, \cdots, u_{|Q|}^Q, \texttt{[SEP]}) \end{aligned} \quad (4)$$

and for each retrieved sentence $D_i^{\mathrm{R}} = (u_1^{D_i^{\mathrm{R}}}, u_2^{D_i^{\mathrm{R}}}, \cdots, u_{|D_i^{\mathrm{R}}|}^{D_i^{\mathrm{R}}})$, in order to let it fully interact with Q when encoding, we concatenate it with Q:

$$\begin{aligned} \mathbf{H}^{D_i^{\mathrm{R}}} &= [\mathbf{h}_1^{D_i^{\mathrm{R}}}; \mathbf{h}_2^{D_i^{\mathrm{R}}}; \cdots; \mathbf{h}_{|D_i^{\mathrm{R}}|+|Q|+3}^{D_i^{\mathrm{R}}}] \\ &= \texttt{BERT}(\texttt{[CLS]}, u_1^{D_i^{\mathrm{R}}}, u_2^{D_i^{\mathrm{R}}}, \cdots, u_{|D_i^{\mathrm{R}}|}^{D_i^{\mathrm{R}}}, \\ &\qquad \texttt{[SEP]}, u_1^Q, u_2^Q, \cdots, u_{|Q|}^Q, \texttt{[SEP]}). \end{aligned} \quad (5)$$

We then concatenate the representation of the question and the representations of all the retrieved sentences:

$$\mathbf{H} = \mathbf{H}^Q \| \mathbf{H}^{D_1^{\mathrm{R}}} \| \cdots \| \mathbf{H}^{D_k^{\mathrm{R}}} \quad (6)$$

where $\|$ represents concatenation.

DSL contains predefined binary operators with numeric arguments (`ADD`, `DIVIDE`, etc.), operators with table row headers as arguments (`TABLE_SUM`, `TABLE_MAX`, etc.) which operate on a row, constant tokens (`CONST_1`, `CONST_100`, `CONST_1000`, etc.) and memory tokens representing the results of previous operations (`#0`, `#1`, etc.).

For all these tokens $U^{\text{DSL}} = \{u_1^{\text{DSL}}, u_2^{\text{DSL}}, \cdots, u_m^{\text{DSL}}\}$ in the DSL, we randomly initialise their representations $\mathbf{H}^{\text{DSL}} = [\mathbf{h}_1^{\text{DSL}}; \mathbf{h}_2^{\text{DSL}}; \cdots; \mathbf{h}_m^{\text{DSL}}]$ in training.

Decoder We begin by introducing the attention mechanism, which will be used frequently in the following:

$$\texttt{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \texttt{Softmax}(\mathbf{W}^Q \mathbf{Q}\mathbf{K}^T)\mathbf{V} \quad (7)$$

where $\mathbf{W}^Q$ is a matrix of learnable parameters.

At step t of generation, the token representation sequence that has been generated is $\mathbf{Y}_t = [\mathbf{y}_1; \mathbf{y}_2; \cdots; \mathbf{y}_{t-1}]$ and the decoder output representation, denoted by $\hat{\mathbf{y}}_t$, is obtained by a LSTM network:

$$\hat{\mathbf{y}}_t = \texttt{LSTM}(\mathbf{y}_{t-1}). \quad (8)$$

We randomly initialize $\mathbf{y}_0$ in training. Our goal is to determine which token in the question, sentences, or DSL should be selected. In the following, unless otherwise stated, all variables are for step t and we may omit the subscript t.

We calculate the attention representation of $\hat{\mathbf{y}}_t$ over $\mathbf{Y}_t$ and $\mathbf{H}^{\text{DSL}} \| \mathbf{H}$ to get a decoder history representation and a sequence representation, respectively:

$$\begin{aligned} \mathbf{a}^{\text{Y}} &= \texttt{Attention}(\mathbf{Q} = \hat{\mathbf{y}}_t, \mathbf{K} = \mathbf{Y}_t, \mathbf{V} = \mathbf{Y}_t) \\ \mathbf{a}^{\text{seq}} &= \texttt{Attention}(\mathbf{Q} = \hat{\mathbf{y}}_t, \mathbf{K} = \mathbf{H}^{\text{DSL}} \| \mathbf{H}, \\ &\qquad \mathbf{V} = \mathbf{H}^{\text{DSL}} \| \mathbf{H}). \end{aligned} \quad (9)$$

Note that $\hat{\mathbf{y}}_t$, as the query of attention, is not included in $\mathbf{a}^{\text{Y}}$ and $\mathbf{a}^{\text{seq}}$, so we fuse them to get a context representation:

$$\mathbf{h}^{\text{c}} = \texttt{Layernorm}(\mathbf{W}^{\text{c}}(\mathbf{a}^{\text{Y}} \| \mathbf{a}^{\text{seq}} \| \hat{\mathbf{y}}_t)) \quad (10)$$

where $\mathbf{W}^{\text{c}}$ is a matrix of learnable parameters. For now, we have obtained the representations of all tokens in DSL, i.e., $\mathbf{H}^{\text{DSL}}$, and all tokens in Q and $\mathbb{D}^{\text{R}}$, i.e., $\mathbf{H}$. We fuse them and compute the scores of all tokens by calculating the inner product of token representation and context representation:

$$\begin{aligned} \mathbf{H}^{\text{S}} &= \mathbf{W}^{\text{S}}((\mathbf{H}^{\text{DSL}} \| \mathbf{H}) \| ((\mathbf{H}^{\text{DSL}} \| \mathbf{H}) \circ \mathbf{a}^{\text{seq}})) \\ \mathbf{s}^{\text{ctx}} &= \mathbf{H}^{\text{S}} \mathbf{h}^{\text{c}} \end{aligned} \quad (11)$$

in which $\circ$ is element-wise product and $\mathbf{W}^{\text{S}}$ is a matrix of learnable parameters. Note that in Equation (11) we duplicate the vector $\mathbf{a}^{\text{seq}}$ to form a matrix to be concatenated with other matrices, which for clarity is omitted in the equation.

We get the reranker scores $\mathbf{s}^{\text{r}}$ from our reranker which will be described in detail in Section 2.3. The final scores of all tokens are represented by the element-wise product of context scores and reranker scores:

$$\mathbf{s} = \mathbf{s}^{\text{ctx}} \circ \mathbf{s}^{\text{r}}. \quad (12)$$

We optimize the cross-entropy loss during training:

$$\mathcal{L} = -\log \frac{\exp(s^{\text{gold}})}{\sum_{tok \in U^{\text{DSL}} \cup U^{\text{num}}} \exp(s_{tok})} \quad (13)$$

where U^{num} is the set of all numeric tokens, s_{tok} is the final score of a token computed by Equation (12), and s^{gold} is the final score of the ground truth token.

We finally select the predicted token by $\mathbf{s}$:

$$\mathbf{y}_t = \mathbf{H}^{\text{S}}_{\arg\max \mathbf{s}} \quad (14)$$

i.e., the representation of the $(\arg\max \mathbf{s})$-th token in $\mathbf{H}^{\text{S}}$. We use teacher forcing mechanism during training, i.e., we use the representation of the ground truth token as $\mathbf{y}_t$.

2.3 Reranker

In this module, we aim to find the sentence that is of interest to the generator at step t, i.e., the one in which the next numeric argument of the arithmetic expression is located.

Reranker Space The reranker space, i.e., the objects that need to be reranked, consists of representations of the question and retrieved sentences.

We construct the reranker space as follows. We use dual attention (Liu et al. 2020) to enhance the interaction between the question representation $\mathbf{H}^Q$ and each sentence representation $\mathbf{H}^{D_i^{\text{R}}}$ computed by Equation (5):

$$\hat{\mathbf{H}}^{Q_i}, \hat{\mathbf{H}}^{D_i^{\text{R}}} = \texttt{DualAttention}(\mathbf{H}^Q, \mathbf{H}^{D_i^{\text{R}}}). \quad (15)$$

$\hat{\mathbf{H}}^{Q_i}$ is the token sequence representations of Q after interacting with $D_i^{\text{R}} \in \mathbb{D}^{\text{R}}$. Then mean-pooling in token dimension is applied to obtain the representations of Q and D_i^{R}:

$$\begin{aligned} \mathbf{h}^{Q_i} &= \texttt{MeanPooling}(\hat{\mathbf{H}}^{Q_i}) \\ \mathbf{h}^{D_i^{\text{R}}} &= \texttt{MeanPooling}(\hat{\mathbf{H}}^{D_i^{\text{R}}}) \end{aligned} \quad (16)$$

We combine the k representations of Q over k sentences to obtain a representation of Q by attention mechanism:

$$\begin{aligned} \mathbf{h}^Q = \texttt{Attention}(\mathbf{Q} = \hat{\mathbf{y}}_t, \mathbf{K} = [\mathbf{h}^{Q_1}; \mathbf{h}^{Q_2}; \cdots, \mathbf{h}^{Q_k}], \\ \mathbf{V} = [\mathbf{h}^{Q_1}; \mathbf{h}^{Q_2}; \cdots, \mathbf{h}^{Q_k}]). \end{aligned} \quad (17)$$

The reranker space contains $\mathbf{h}^Q, \mathbf{h}^{D_1^{\text{R}}}, \ldots, \mathbf{h}^{D_k^{\text{R}}}$.

Reranker Query The query used to rerank is dynamically generated, not only in relation to the expression that has been generated, i.e., $\mathbf{Y}_t$, but also in relation to the source sentences from which the tokens in the expression are extracted. We first locate the source sentence of each token in the generated expression. We take the representation of its source sentence if the generated token is numeric, or keep the representation of the token if it is from the DSL, formally:

$$\texttt{source}(\mathbf{y}_i) = \begin{cases} \mathbf{y}_i, & \text{if } \mathbf{y}_i \in \mathbf{H}^{\text{DSL}}, \\ \mathbf{h}^D, & \text{if } \mathbf{y}_i \text{ comes from } D \in \mathbb{D}^{\text{QR}} \end{cases} \quad (18)$$

where $\mathbb{D}^{\text{QR}} = \{Q\} \cup \mathbb{D}^{\text{R}}$ and $\mathbf{h}^D$ refers to $\mathbf{h}^Q$ computed by Equation (17) or $\mathbf{h}^{D_i^{\text{R}}}$ computed by Equation (16). We obtain the source sentence representations of $\mathbf{Y}_t$ by

$$\mathbf{H}_t^{\text{src}} = \mathbf{H}_{t-1}^{\text{src}}; \texttt{source}(\mathbf{y}_{t-1}) \quad (19)$$

which refers to appending the source sentence representation of $\mathbf{y}_{t-1}$ to the source sentence representations of $\mathbf{Y}_{t-1}$ which were calculated at the previous step.

We use the source sentence representations of generated tokens to construct a query to rerank all the sentences at the current step. We compute the attention representation of $\hat{\mathbf{y}}_t$ over $\mathbf{H}_t^{\text{src}}$ to obtain the reranker query representation $\mathbf{Q}^{\text{s}}$:

$$\mathbf{Q}^{\text{s}} = \texttt{Attention}(\mathbf{Q} = \hat{\mathbf{y}}_t, \mathbf{K} = \mathbf{H}_t^{\text{src}}, \mathbf{V} = \mathbf{H}_t^{\text{src}}). \quad (20)$$

Reranker Scorer With the reranker query $\mathbf{Q}^{\mathrm{s}}$ and the reranker space $\{\mathbf{h}^{Q}, \mathbf{h}^{D_1}, \cdots, \mathbf{h}^{D_k}\}$, we re-compute the relevance score of each sentence by:

$$
\begin{aligned}
s^{Q} &= \texttt{Linear}(\texttt{Tanh}(\texttt{Linear}([\mathbf{Q}^{\mathrm{s}} \| \mathbf{h}^{Q}]))) \\
s^{D_i^{\mathrm{R}}} &= \texttt{Linear}(\texttt{Tanh}(\texttt{Linear}([\mathbf{Q}^{\mathrm{s}} \| \mathbf{h}^{D_i^{\mathrm{R}}}]))), i \in \{1, \cdots, k\}
\end{aligned} \tag{21}
$$

The reranker scores in Equation (12) are given by

$$
\mathbf{s}^{\mathrm{r}} = \texttt{Expand}(\texttt{Softmax}([s^{Q}, s^{D_1^{\mathrm{R}}}, \cdots, s^{D_k^{\mathrm{R}}}])). \tag{22}
$$

Note that $s^{Q}, s^{D_1^{\mathrm{R}}}, \cdots, s^{D_k^{\mathrm{R}}}$ are for scoring sentences while we need to obtain reranker scores for tokens, so an `Expand` operation is applied which assigns the score of each sentence to every token in that sentence. The scores of tokens in the DSL are fixed to 1.0.

3 Experiments

3.1 Datasets

To the best of our knowledge, there were two datasets requiring generating arithmetic expressions for numerical reasoning over tabular and textual data: FinQA (Chen et al. 2021b) and MultiHiertt (Zhao et al. 2022). We used FinQA in our experiments. We did not use MultiHiertt because it was new and yet to be further polished after contacting its authors.

FinQA contains 8,281 financial questions which were collected from financial reports of S&P 500 companies. Questions were divided into 6,251 (75%) for training, 883 (10%) for development, and 1,147 (15%) for testing. Each question contains a table where the average number of rows is 6.36. The average number of sentences accompanying a question is 24.32. The total number of tokens in the table and sentences accompanying a question averages 687.53, with a maximum value of 2,679.

3.2 Implementation Details

We experimented on NVIDIA V100 (32GB) GPUs and Ubuntu 18.04. Our implementations were based on PyTorch 1.11. We selected models on the development set.

For the encoder in our retriever, we used BERT-base-uncased with hidden layer $= 12$, hidden units $= 768$, and attention heads $= 12$. We set epoch $= 8$ and seed $= 8$. We used the Adam optimizer with learning rate $= 2e-5$ selected from $\{1.5e-5, 2e-5\}$ with batch size $= 16$. We set $k = 3$ and max sequence length $= 256$.

For the encoder in our generator, we used two pretrained models: BERT-base-uncased and RoBERTa-large. RoBERTa-large is with hidden layer $= 24$, hidden units $=$ 1,024, and attention heads $= 16$. In our reranker and generator, we set epoch $= 300$ and seed $= 8$. We used the Adam optimizer with learning rate $= 2e-5$ for BERT-base-uncased and learning rate $= 1.5e-5$ for RoBERTa-large, both selected from $\{7e-6, 1.5e-5, 2e-5\}$. We set batch size $= 16$ for BERT-base-uncased and batch size $=$ 24 for RoBERTa-large, both selected from $\{16, 24, 32\}$. We set max sequence length $= 256$.

	Dev		Test	
Methods	EA	PA	EA	PA
Longformer$_{\texttt{base}}$	23.83	22.56	21.90	20.48
NeRd	47.53	45.37	48.57	46.76
FinQANet$_{\texttt{FinBERT}}$	46.64	44.11	50.10	47.52
FinQANet$_{\texttt{BERT}}$	49.91	47.15	50.00	48.00
FinQANet$_{\texttt{RoBERTa}}$	61.22	58.05	61.24	58.86
DyRRen$_{\texttt{BERT}}$	61.16	58.32	59.37	57.54
DyRRen$_{\texttt{RoBERTa}}$	**66.82**	**63.87**	**63.30**	**61.29**
Human Expert	–	–	91.16	87.49
General Crowd	–	–	50.68	48.17

Table 1: Comparison with baselines on FinQA. `BERT` refers to BERT-base-uncased, `RoBERTa` refers to RoBERTa-large and `FinBERT` is a BERT-like model pretrained in the financial domain.

3.3 Baselines

We compared our proposed model DyRRen with known methods and results reported in the literature.

We mainly compared DyRRen with FinQANet, which is a state-of-the-art retriever-generator model specifically designed for numerical reasoning over tabular and textual data. We compared with FinQANet's versions on FinBERT (Araci 2019), BERT-base-uncased, and RoBERTa-large. Chen et al. (2021b) provided the results of all the three versions on FinQA.

Longformer (Beltagy, Peters, and Cohan 2020) is specifically designed to handle long documents, and was used in our experiments to verify the necessity to include a retriever in our framework. NeRd (Chen et al. 2020c) is an expression generator based on a pointer network and has achieved competitive results on MathQA (Amini et al. 2019) and DROP (Dua et al. 2019). We used the results of Longformer and NeRd reported in Chen et al. (2021b).

3.4 Evaluation Metrics

Following the literature, we measured program accuracy (PA), i.e., proportion of correctly generated arithmetic expressions, and execution accuracy (EA), i.e., proportion of correct final answers. Note that sometimes a generated expression is different from the corresponding ground truth but their answers are equivalent, so execution accuracy is always at least as high as program accuracy.

3.5 Main Results

On the test set, as shown in Table 1, DyRRen outperformed all the baselines by at least 9.37% of EA and 9.54% of PA with BERT and at least 2.06% of EA and 2.43% of PA with RoBERTa. The differences were statistically significant under $p < 0.01$. Besides, both DyRRen$_{\texttt{BERT}}$ and DyRRen$_{\texttt{RoBERTa}}$ exceeded general crowd (50.68% of EA and 48.17% of PA), although they were still not comparable with human expert (91.16% of EA and 87.49% of PA).

It is notable that DyRRen$_{\texttt{BERT}}$ achieved similar performance to FinQANet$_{\texttt{RoBERTa}}$, while the number of parameters in FinQANet$_{\texttt{RoBERTa}}$ ($\sim$380M) is even 2.7 times that of DyRRen$_{\texttt{BERT}}$ ($\sim$139M).

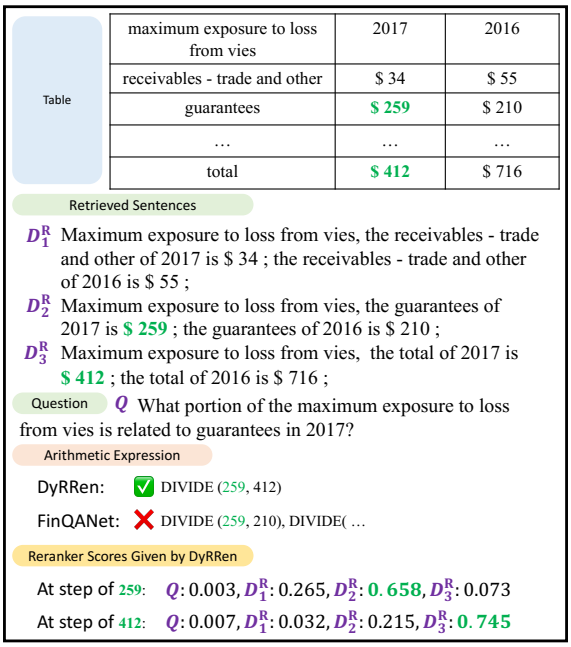

Figure 4: Case study sampled from FinQA.

3.6 Case Study

As shown in Figure 4, we sampled a question from FinQA for which the retrievers of both DyRRen and FinQANet (BERT versions) successfully retrieved the sentences relevant to the question, i.e., D_2^R and D_3^R. It can be seen that at the beginning, both DyRRen and FinQANet succeeded in generating the correct operator `DIVIDE` and extracting the first argument "259" from D_2^R, and our reranker in DyRRen exhibited the highest confidence to D_2^R (0.658). When generating the second argument, our reranker assigned a higher score (0.745) to D_3^R containing the correct argument "412", successfully switching the generator's attention to the correct sentence, while FinQANet still stayed in D_2^R and selects "210", thus failing to answer this question.

3.7 Ablation Study

We compared DyRRen with two variants:

- DyRRen$_{\texttt{no-reranker}}$, which removes the reranker module from our retrieval-ranker-generator framework.
- DyRRen$_{\texttt{vanilla-retriever}}$, which replaces our retriever with FinQANet's retriever to verify the effectiveness of the optimizations we have made to our retriever.

On FinQA, as shown in Table 2, DyRRen$_{\texttt{no-reranker}}$ was 1.48% lower in EA and 2.00% lower in PA than DyRRen on the basis of BERT on the test set, demonstrating the usefulness of our reranker. The performance of DyRRen$_{\texttt{no-reranker}}$ (RoBERTa version) also noticeably dropped on the dev set (by 0.98% in EA and 1.02% in PA), while the differences on the test set were neglectable. We thought it was the more powerful RoBERTa which counterbalanced some gain from our reranker.

Methods	Dev		Test	
	EA	PA	EA	PA
BERT				
DyRRen	61.16	58.32	59.37	57.54
DyRRen$_{\texttt{no-reranker}}$	60.82	58.66	57.89	55.54
DyRRen$_{\texttt{vanilla-retriever}}$	58.78	56.17	56.15	53.79
RoBERTa				
DyRRen	66.82	63.87	63.30	61.29
DyRRen$_{\texttt{no-reranker}}$	65.80	62.85	62.86	61.46
DyRRen$_{\texttt{vanilla-retriever}}$	62.85	60.48	60.33	57.89

Table 2: Ablation study on FinQA.

The performance of DyRRen$_{\texttt{vanilla-retriever}}$ decreased on both BERT and RoBERTa based experiments. For example, DyRRen$_{\texttt{vanilla-retriever}}$ (BERT) was 3.22% lower in EA and 3.75% lower in PA than DyRRen on the test set, demonstrating the effectiveness of our retriever.

We also tried to incorporate models pre-trained on tables such as TaPas (Herzig et al. 2020) into our encoder, but did not obtain better results.

3.8 Fine-Grained Results

We conducted fine-grained comparison between DyRRen$_{\texttt{BERT}}$ and FinQANet$_{\texttt{BERT}}$, as shown in Table 3.

Questions Categorized by Question Type Our model made notable improvements over FinQANet on all the three types of questions. We also observed that both models performed the best on table-only questions, i.e., those that can be answered only based on tables. This may be due to the fact that row sentences have good structure as they are converted by using templates, so they are relatively easy to be understood. Both models performed the worst on table-sentence questions, indicating that there is still room for improving hybrid data understanding.

Questions Categorized by Expression Length Our reranker helped our model focus on the unique supporting sentence for the current step rather than on all the retrieved sentences, and it was consistently superior on questions of all expression lengths. We also found that both models performed poorly when expression length exceeded two. These "long-distance reasoning" questions are very challenging for existing methods and await further study.

3.9 Error Analysis

We randomly sampled 50 questions for which our model failed to generate the correct arithmetic expression. The errors caused by our retriever constituted about 20% of all errors, e.g., the retriever failed to retrieve all the supporting sentences. About 14% of the errors were caused by the reranker misleading the generator, i.e., the reranker assigned improper scores to sentences in some generation step. The rest of the errors (66%) were due to the generator. There were many reasons for these generation errors, one of the main ones being that the generator failed to actively switch attention between sentences, and the failure was too strong to be counterbalanced by our reranker. Other reasons include incorrect prediction of operators, or the generator became

	Dev		Test	
	EA	PA	EA	PA
DyRRen$_{\text{BERT}}$				
Question 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 length				
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
FinQANet$_{\text{BERT}}$				
Question type				
table-only	61.62	58.86	58.22	55.67
sentence-only	37.37	35.86	40.64	39.93
table-sentence	28.67	27.27	31.01	30.38
Expression length				
1	52.77	51.05	52.14	50.76
2	54.01	50.52	53.55	51.10
> 2	24.66	23.29	17.86	15.48

Table 3: Fine-grained comparison with FinQANet on FinQA by question type and expression length.

lost on questions that require multi-step numerical reasoning or require external financial knowledge.

4 Related Work

4.1 Question Answering over Hybrid Data

QA that requires reasoning over a hybrid context, i.e., tabular and textual data, is a task that has emerged in the last two years. Two of the first datasets focusing on this task are HybridQA (Chen et al. 2020b) and TSQA (Li, Sun, and Cheng 2021). The former is an extractive QA dataset focusing more on multi-hop reasoning while the latter is a multiple-choice QA dataset more focusing on numerical reasoning. Eisenschlos et al. (2021) proposes a Transformer-based model with row-wise and column-wise attention and (Kumar et al. 2021) applies multiple-instance training to solve HybridQA. Li, Sun, and Cheng (2021) designs a number of templates containing numerical operations to convert tables to sentences, and then uses a knowledge-injected pretrained language model to solve TSQA.

Following these efforts, a growing number of more challenging hybrid QA datasets have emerged. OTT-QA (Chen et al. 2021a) and NQ-Tables (Herzig et al. 2021) place hybrid QA in an open-domain setting where tables are not given but need to be retrieved from a table corpus. Herzig et al. (2021) solves this problem by dense retrieval. Later, it was found that tables in the financial domain carry a lot of numerical content and are suitable for QA tasks that require numerical reasoning over tabular and textual data. TAT-QA (Zhu et al. 2021), FinQA (Chen et al. 2021b), and MultiHiertt (Zhao et al. 2022) are such datasets containing QA pairs from financial reports. TAT-QA requires the answer to a question to be extracted or calculated from given sentences and tables. FinQA and MultiHiertt require the generation of an arithmetic expression, making the question answering process more interpretable.

FinQANet (Chen et al. 2021b), which adopts a retrieval-generator architecture, remains state of the art for solving hybrid QA that requires the generation of arithmetic expressions. The major architectural difference between FinQANet and our DyRRen is that we fuse the idea of reranking into each generation step, i.e., our dynamic reranker, to enhance our generator. There are also retrieval-free methods (Lei et al. 2022) which are orthogonal to our work.

4.2 Numerical Reasoning in QA

While current pretrained language models (Lin et al. 2020) could not perform well on numerical reasoning, there is a growing interest in numerical reasoning based QA. DROP (Dua et al. 2019) requires mathematical operations such as adding, subtracting, counting, and sorting. Later work shows that it can be solved by semantic parsing (Gupta et al. 2020), graph-base method (Chen et al. 2020a), neural module network (Zhou et al. 2022), or enhancing number-related context understanding (Kim et al. 2022).

Math word problem (MWP) (Hosseini et al. 2014; Huang et al. 2016; Amini et al. 2019) is a fundamental and challenging task which has been introduced for years. The task is to, given a short textual question, calculate the answer by generating an arithmetic expression. There are two major differences between MWP and the task considered in this paper: 1) MWP contains only a short textual question without tables and supporting sentences, so solving MWPs does not require to handle tabular data nor the retrieval problem. 2) Questions in MWPs are described in a controlled manner, exhibiting strong regularity so some of the early work could use statistical learning (Hosseini et al. 2014; Kushman et al. 2014) or rule/template-based methods (Shi et al. 2015; Wang et al. 2019) to achieve fairly good results. Huang et al. (2021) proposes a recall learning framework for solving MWPs, a framework that recalls problems with similar expression structures, benefiting from the regularity of MWPs. There are also graph/tree based (Wu, Zhang, and Wei 2021; Jie, Li, and Lu 2022) or multiple-task learning (Qin et al. 2021; Shen et al. 2021) methods. However, due to the above-mentioned differences between two tasks, these solutions to MWPs are unlikely to obtain promising results on our task.

5 Conclusion

We present DyRRen, a dynamic retriever-reranker-generator model for numerical reasoning over tabular and textual data. We believe that the idea of incorporating a reranker to enhance generation has the potential to be applicable to other generation tasks where the input contains multiple sentences, including machine translation, machine writing, summarization, etc., which we will explore as future work.

Our error analysis also reveals some limitations of our model. Our reranker is able to help correct the selection of numerical arguments of operators, but is unable to handle incorrect prediction of operators. We will extend our reranker to handle operators in future work. We are also ready to experiment on more datasets to evaluate the generalizability of our model, for instance, when the emerging MultiHiertt dataset becomes stable.

Acknowledgments

This work was supported in part by the NSFC (62072224) and in part by the CCF-Baidu Open Fund.

References

Amini, A.; Gabriel, S.; Lin, S.; Koncel-Kedziorski, R.; Choi, Y.; and Hajishirzi, H. 2019. MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 2357–2367.

Araci, D. 2019. FinBERT: Financial Sentiment Analysis with Pre-trained Language Models. *CoRR*, abs/1908.10063.

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The Long-Document Transformer. *CoRR*, abs/2004.05150.

Burges, C. J. 2010. From RankNet to LambdaRank to LambdaMART: An Overview. Technical Report MSR-TR-2010-82, Microsoft Research.

Chen, K.; Xu, W.; Cheng, X.; Xiaochuan, Z.; Zhang, Y.; Song, L.; Wang, T.; Qi, Y.; and Chu, W. 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 2020, Online, November 16-20, 2020*, 6759–6768.

Chen, W.; Chang, M.; Schlinger, E.; Wang, W. Y.; and Cohen, W. W. 2021a. Open Question Answering over Tables and Text. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

Chen, W.; Zha, H.; Chen, Z.; Xiong, W.; Wang, H.; and Wang, W. Y. 2020b. HybridQA: A Dataset of Multi-Hop Question Answering over Tabular and Textual Data. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020, 1026–1036.

Chen, X.; Liang, C.; Yu, A. W.; Zhou, D.; Song, D.; and Le, Q. V. 2020c. 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*.

Chen, Z.; Chen, W.; Smiley, C.; Shah, S.; Borova, I.; Langdon, D.; Moussa, R.; Beane, M.; Huang, T.; Routledge, B. R.; and Wang, W. Y. 2021b. FinQA: A Dataset of Numerical Reasoning over Financial Data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, 3697–3711.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 4171–4186.

Dua, D.; Wang, Y.; Dasigi, P.; Stanovsky, G.; Singh, S.; and Gardner, M. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 2368–2378.

Eisenschlos, J.; Gor, M.; Müller, T.; and Cohen, W. W. 2021. MATE: Multi-view Attention for Table Transformer Efficiency. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, 7606–7619.

Gupta, N.; Lin, K.; Roth, D.; Singh, S.; and Gardner, M. 2020. Neural Module Networks for Reasoning over Text. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Herzig, J.; Müller, T.; Krichene, S.; and Eisenschlos, J. 2021. Open Domain Question Answering over Tables via Dense Retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, 512–519.

Herzig, J.; Nowak, P. K.; Müller, T.; Piccinno, F.; and Eisenschlos, J. M. 2020. TaPas: Weakly Supervised Table Parsing via Pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, 4320–4333.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8): 1735–1780.

Hosseini, M. J.; Hajishirzi, H.; Etzioni, O.; and Kushman, N. 2014. Learning to Solve Arithmetic Word Problems with Verb Categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 523–533.

Huang, D.; Shi, S.; Lin, C.; Yin, J.; and Ma, W. 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, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Huang, S.; Wang, J.; Xu, J.; Cao, D.; and Yang, M. 2021. Recall and Learn: A Memory-augmented Solver for Math Word Problems. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, 786–796.

Jie, Z.; Li, J.; and Lu, W. 2022. Learning to Reason Deductively: Math Word Problem Solving as Complex Relation Extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1:*

Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022, 5944–5955.

Khattab, O.; and Zaharia, M. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, 39–48.

Kim, J.; Kang, J.; Kim, K.; Hong, G.; and Myaeng, S. 2022. Exploiting Numerical-Contextual Knowledge to Improve Numerical Reasoning in Question Answering. In *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, 1811–1821.

Kumar, V.; Chemmengath, S. A.; Gupta, Y.; Sen, J.; Bharadwaj, S.; and Chakrabarti, S. 2021. Multi-Instance Training for Question Answering Across Table and Linked Text. *CoRR*, abs/2112.07337.

Kushman, N.; Zettlemoyer, L.; Barzilay, R.; and Artzi, Y. 2014. Learning to Automatically Solve Algebra Word Problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, 271–281.

Lei, F.; He, S.; Li, X.; Zhao, J.; and Liu, K. 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, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, 1379–1390.

Li, X.; Sun, Y.; and Cheng, G. 2021. TSQA: Tabular Scenario Based Question Answering. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 13297–13305.

Lin, B. Y.; Lee, S.; Khanna, R.; and Ren, X. 2020. Birds have four legs?! NumerSense: Probing Numerical Commonsense Knowledge of Pre-Trained Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, 6862–6868.

Liu, D.; Gong, Y.; Fu, J.; Yan, Y.; Chen, J.; Jiang, D.; Lv, J.; and Duan, N. 2020. RikiNet: Reading Wikipedia Pages for Natural Question Answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, 6762–6771.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*, abs/1907.11692.

Qin, J.; Liang, X.; Hong, Y.; Tang, J.; and Lin, L. 2021. Neural-Symbolic Solver for Math Word Problems with Auxiliary Tasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, 5870–5881.

Shen, J.; Yin, Y.; Li, L.; Shang, L.; Jiang, X.; Zhang, M.; and Liu, Q. 2021. Generate & Rank: A Multi-task Framework for Math Word Problems. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, 2269–2279.

Shi, S.; Wang, Y.; Lin, C.; Liu, X.; and Rui, Y. 2015. Automatically Solving Number Word Problems by Semantic Parsing and Reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, 1132–1142.

Wang, L.; Zhang, D.; Zhang, J.; Xu, X.; Gao, L.; Dai, B. T.; and Shen, H. T. 2019. Template-Based Math Word Problem Solvers with Recursive Neural Networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 7144–7151.

Wu, Q.; Zhang, Q.; and Wei, Z. 2021. An Edge-Enhanced Hierarchical Graph-to-Tree Network for Math Word Problem Solving. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, 1473–1482.

Zhan, J.; Mao, J.; Liu, Y.; Guo, J.; Zhang, M.; and Ma, S. 2021. Optimizing Dense Retrieval Model Training with Hard Negatives. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, 1503–1512.

Zhao, Y.; Li, Y.; Li, C.; and Zhang, R. 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), ACL 2022, Dublin, Ireland, May 22-27, 2022*, 6588–6600.

Zhou, Y.; Bao, J.; Duan, C.; Sun, H.; Liang, J.; Wang, Y.; Zhao, J.; Wu, Y.; He, X.; and Zhao, T. 2022. 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, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, 1655–1666.

Zhu, F.; Lei, W.; Huang, Y.; Wang, C.; Zhang, S.; Lv, J.; Feng, F.; and Chua, T. 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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, 3277–3287.