Doc2SoarGraph: Discrete Reasoning over Visually-Rich Table-Text Documents via Semantic-Oriented Hierarchical Graphs

Fengbin Zhu[1], Chao Wang[2], Fuli Feng[3*], Zifeng Ren[1], Moxin Li[1], Tat-Seng Chua[1]

[1]National University of Singapore, [2]6Estates Pte Ltd, [3]University of Science and Technology of China
{zhfengbin, fulifeng93}@gmail.com, wangchao@6estates.com

Abstract

Table-text document (e.g., financial reports) understanding has attracted increasing attention in recent two years. TAT-DQA (Zhu et al., 2022) is a realistic setting for the understanding of visually-rich table-text documents, which involves answering associated questions requiring discrete reasoning. Most existing work relies on token-level semantics, falling short in the reasoning across document elements such as quantities and dates. To address this limitation, we propose a novel **Doc2SoarGraph** model that exploits element-level semantics and employs **S**emantic-**o**riented hier**ar**chical **Graph** structures to capture the differences and correlations among different elements within the given document and question. Extensive experiments on the TAT-DQA dataset reveal that our model surpasses the state-of-the-art conventional method (i.e., MHST) and large language model (i.e., ChatGPT) by 17.73 and 6.49 points respectively in terms of Exact Match (EM) metric, demonstrating exceptional effectiveness. The source code is publicly available at https://github.com/fengbinzhu/Doc2SoarGraph/.

Keywords: Visually-rich table-text document, Question answering, Discrete reasoning, FinTech

1. Introduction

Table-text documents containing a hybrid of tabular and textual data are pervasive in the real world, e.g. SEC filings, academic papers and medical reports. Recently, there has been a surge of work attempting to intelligently understand table-text documents through answering associated questions (Chen et al., 2020; Zhu et al., 2021; Chen et al., 2021; Zhao et al., 2022). However, these works focus on the well-annotated structured tables and manually selected paragraphs from the original documents, which is not in line with reality.

Research on the intelligent understanding of real-world table-text documents has been activated with the release of TAT-DQA (Zhu et al., 2022), a Document Visual Question Answering (DocVQA) challenge over financial documents. In TAT-DQA, each document contains extensive numerical data in both tabular and textual formats, where discrete reasoning capabilities (*e.g., arithmetic calculation, comparison, counting and sorting*) are demanded to answer the questions. One example is shown in Figure 1. To address this challenge, MHST (Zhu et al., 2022) applies sequence tagging on each token to select relevant tokens from the document, followed by answer inference over the selected tokens. Though effective, the performance of MHST is still not optimal. One reason is that the tokens only carry part of the semantics of the original data. For example, as shown in Figure 1, the spectrum license fee in 2019 is $1,731$ million, while the quantity $1,731$ corresponds to four tokens, i.e., "$\underline{1}$", "$\underline{,}$", "$\underline{73}$", and "$\underline{\#\#1}$" after tokenization. The model can

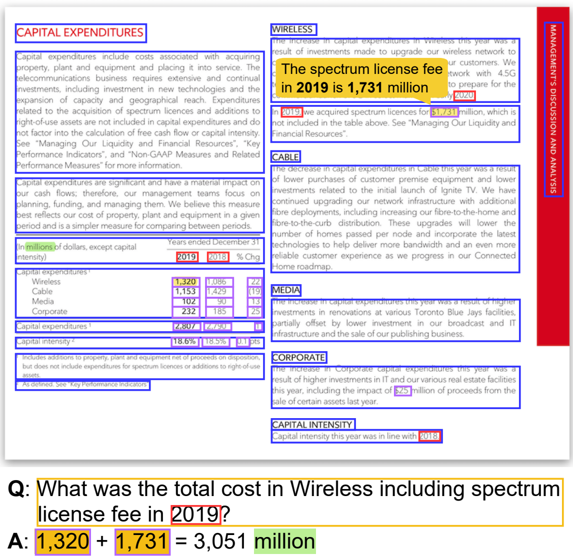

Figure 1: An example from TAT-DQA. We leverage four types of semantic elements from the question and document to facilitate discrete reasoning, i.e., *Date*, *Quantity*, *Question* and *Block*, marked in red, purple, yellow and blue rectangle, respectively. The quantities with yellow background are supporting evidence to the question. The "million" with green background is the scale of the answer.

hardly infer the meaning of the original number from every single token unless they are all combined.

To mitigate this issue, we exploit element-level semantics to facilitate discrete reasoning. As shown in Figure 1, we consider four types of elements, including *Question*, *Block*, *Quantity* and *Date*. Each

*Corresponding author

LREC-COLING 2024, pages 5119–5131

of these elements carries more complete semantics than single tokens that can be leveraged by the model. The differences and correlations among them can provide rich and crucial clues for the model to conduct reasoning to derive the answer. For example, though 2019 and $1,731$ in Figure 1 are both numerical values, the former refers to "year 2019" (date), while the latter is "spectrum license fee" (quantity), which cannot be compared. As such, it would be more appropriate to model the different types of elements separately. Moreover, to understand the numerical value $1,731$ in the document, it is essential to consider the text information of the corresponding document block. Thus, the correlations of different elements should also be leveraged to facilitate model's reasoning process.

In this work, we propose a **Doc2SoarGraph** model for question answering over visually-rich table-text **doc**uments with **s**emantic-**o**riented hier**ar**chical **graph**s. It models the differences and correlations of the elements (i.e., quantities, dates, question and document blocks) in the input data with hierarchy graph structures taking each element as one node. Considering that about 20% of the documents are multi-page, we first transform each multi-page document to a single image of the model preferred dimension. Then, given a question and a document, we adopt LayoutLMv2 (Xu et al., 2021) to take in the question, document text and the corresponding layout and document image, and initializes the representations of all semantic elements with the output. After that, we construct a hierarchy of four graphs in two levels. In the first level, we build three graphs: a Quantity Comparison (QC) graph to model the magnitude and comparison among all the *Quantity* nodes; a Date Comparison (DC) graph to model the time sequence among all the *Date* nodes; a Text Relation (TR) graph with the *Question* node and *Block* nodes as these nodes usually contain rich text information. In the second level, on top of these three graphs, a Semantic Dependency (SD) graph is built with all types of nodes to model the semantic relationships and dependencies among them. Then, the model selects the most question-relevant nodes from the SD graph and applies different reasoning strategies over the selected nodes to derive the final answer based on the answer type.

Our main contributions are three-fold. 1) We propose to exploit element-level semantics to facilitate discrete reasoning over visually-rich table-text documents. 2) We develop a novel Doc2SoarGraph model to model the differences and correlations among various elements with semantic-oriented hierarchical graph structures, which owns greatly enhanced evidence extraction and discrete reasoning capabilities. 3) We conduct extensive experiments on TAT-DQA dataset, and the results show that our Doc2SoarGraph model outperforms both state-of-the-art conventional method (i.e., MHST) and large language model (LLMs) (i.e., ChatGPT) by 17.73 and 6.46 points respectively in Exact Match (EM), demonstrating remarkable effectiveness.

2. Doc2SoarGraph Model

Consider a natural language question denoted as Q, and a visually-rich table-text document denoted as D with several pages $P = (P_1, P_2, ..., P_{|P|})$, where $|P|$ is the number of pages. In the document D, the page p has a list of blocks $B^p = (B^p_1, B^p_2, ..., B^p_{|B|})$ that are generated by an OCR/PDF converter, where $|B|$ is the number of blocks on the page p. Our goal is to generate the answer to the question Q that usually requires discrete reasoning based on the document D. To solve the problem, we develop a Doc2SoarGraph model. An overall architecture is illustrated in Figure 2.

2.1. Document Transformation

As pre-processing, we transform each multi-page document in TAT-DQA into a one-page document with a simple yet effective method. In particular, we first transform each page to a single image with the same dimension and then combine the corresponding multiple images of the pages vertically following the original page order. Then, we resize the combined image to the dimension of a single-page document, which is preferred by the model. Since the document text and layout information are available in TAT-DQA, we further adjust the layout information according to the dimension of the final document image. After that, all documents are considered as single-page documents and we obtain the initial visual embeddings of each document by applying the same CNN-based encoder as LayoutLMv2 (Xu et al., 2021).

2.2. Node Initialization

Rather than only relying on token-level semantics, our method also exploits element-level semantics to facilitate discrete reasoning with graph structures. In particular, we harness four types of elements, namely, the question, each document block generated by the OCR/PDF converter, each quantity and each date in the question and the document block, which are named *Question*, *Block*, *Quantity* and *Date*, respectively. We take each type of element as a kind of node, and get four types of nodes to build the graphs, i.e., *Question* node, *Block* node, *Quantity* node and *Date* node. We then employ LayoutLMv2$_{\text{LARGE}}$ (Xu et al., 2021) to take as input the question, the document text and layout information, and the final document image, and output

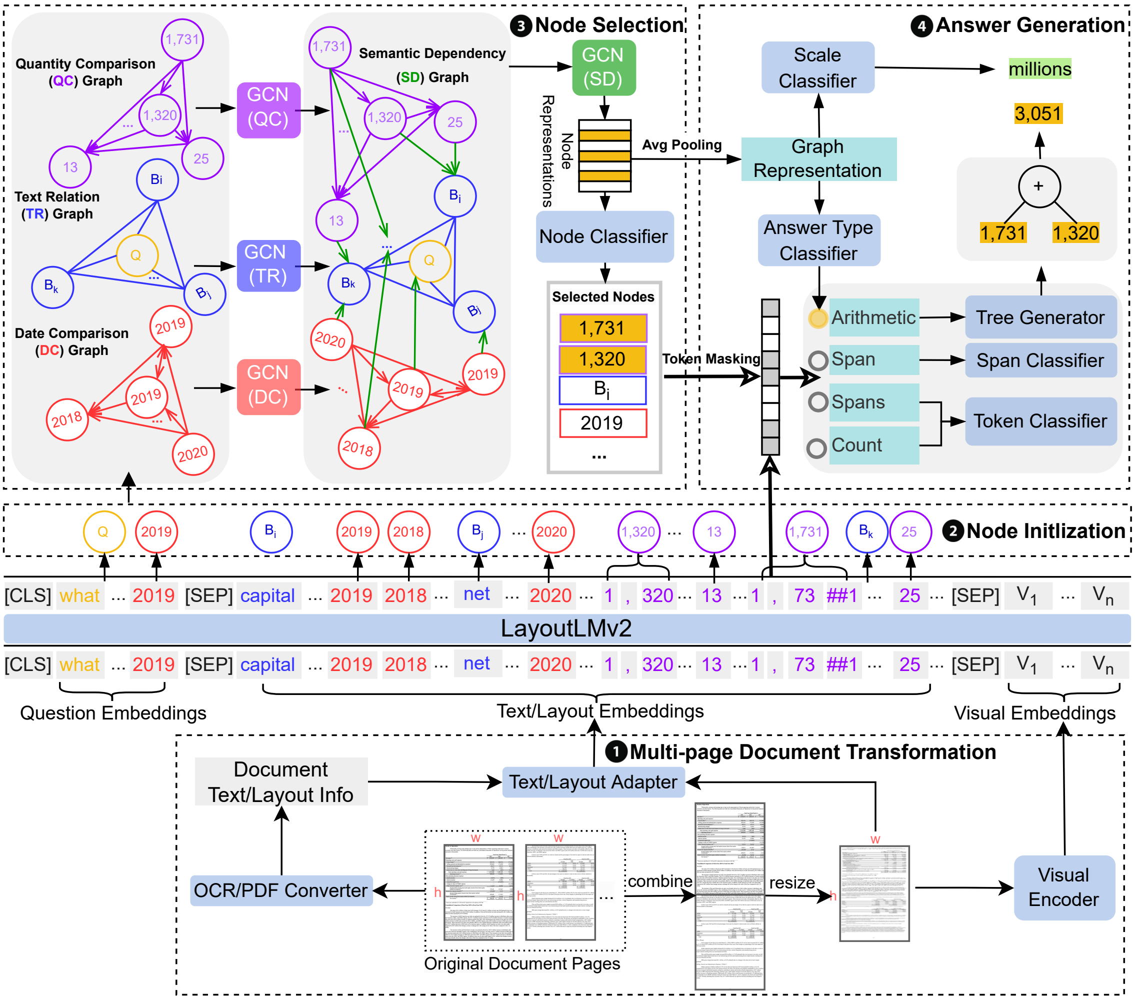

Figure 2: An overview of proposed Doc2SoarGraph model. Take the sample in Figure 1 as an example.

the token-level hidden representations. Then, we compute the mean of the corresponding tokens for each node as its initial representation.

2.3. Node Selection

Based on the four types of nodes as explained above, we construct hierarchical graphs to model their relationships so as to select those most relevant nodes as the supporting evidence to the question and facilitate discrete reasoning of the model.

• **Hierarchical Graphs Construction.** We construct four graphs, which form a two-level hierarchy, to model the element-level semantics. Formally, a graph G is represented by an adjacency matrix A $\in R^{N\times N}$, where N is the number of nodes. If there is an edge connecting the i^{th} and j^{th} nodes, we assign value 1 to the corresponding position (i, j) in the matrix A, and otherwise 0.

Quantity Comparison (QC) Graph (denoted as G_{QC}): It is dedicated to retaining the numerical magnitude and comparison between every two quantities. For two *Quantity* nodes q_i, q_j, if $q_i \geq q_j$, a directed edge e_{ij} = (q_i, q_j) pointing from q_i to q_j is added following NumNet (Ran et al., 2019).

Date Comparison (DC) Graph (denoted as G_{DC}): It is dedicated to retaining the time sequence and comparison between every two dates. For two *Date* nodes d_i, d_j, a directed edge e_{ij} = (d_i, d_j) pointing from d_i to d_j is added if $d_i \geq d_j$ (d_i later than d_j).

Text Relation (TR) Graph (denoted as G_{TR}): It is dedicated to associating the informative descriptions among the question and the document blocks. The *Question* node and a *Block* node or every two *Block* nodes will have an undirected edge between them, forming a fully-connected graph.

Semantic Dependency (SD) Graph (denoted as G_{SD}): It is built with all the four types of nodes to model the semantic dependencies of the *Quantity* or *Date* node upon the *Question* or *Block* node, besides attaining all the correlations in the above three graphs. 1) Edges for two *Quantity* nodes, two *Date* nodes, a *Question* node and a block node, or two *Block* nodes will be added in G_{SD} following the construction rules for G_{QC}, G_{DC}, and G_{TR}, respectively. 2) Between one *Quantity* node and one *Question* or *Block* node, a directed edge pointing from the *Quantity* node to the *Question* node or the *Block* node will be added to the graph G_{SD} if the quantity is part of the question or block; edges between one *Date* node and one *Question* or *Block* node are added in the same way.

• **Node Classifier.** After constructing the hierarchical graphs, a dedicated graph convolution network (GCN) (Kipf and Welling, 2017a) is applied for each graph to learn node representations respectively. As illustrated in Figure 2, the GCN (QC), GCN (DC) and GCN (TR) are applied respectively on the QC graph, DC graph and TR graph to learn corresponding node representations, which are then used to initialize the node representations of the SD graph. The GCN (SD) is applied on the SD graph to learn the final representation of each node h_{node}. A binary node classifier is then applied on each node in the SD graph to predict whether the node is relevant to the question or not. The probability on node classification is computed as

$$\mathrm{P}_{\textsf{node}} = \mathrm{softmax}(\mathrm{FFN}(h_{\mathrm{node}})) \tag{1}$$

where FFN is a feed-forward network with two layers. All the nodes that are classified as relevant to the question are collected. The representation of the SD graph h_{SD} is obtained by computing the mean of all the node representations in SD graph.

2.4. Answer Generation

We generate the final answer with the selected nodes, as follows.

• **Token Masking.** Based on the selected nodes, we mask the tokens that are not included in the selected *Block* nodes to reduce the search space for answer prediction and update the token representations with their corresponding block node representations. Particularly, we obtain the token-level representations from the output of the LayoutLMv2 encoder first. Then, we mask the tokens not covered by any selected block nodes. For tokens that are included in the selected block nodes, we update the representation of each token by concatenating its token representation with the corresponding block representation,

$$h^{'}_{token} = \mathrm{concat}(h_{token}, h_{node}) \tag{2}$$

where h_{token} is the token representation output from the encoder; h_{node} is the representation of the token's corresponding block node obtained from the SD graph; $concat$ denotes concatenation; $h^{'}_{token}$ is the updated token representation. For tokens that are masked in the sequence, we pad their representations with zero. Finally, we obtain a sequence of updated token representations $h^{'}_{[t_1,t_2,\ldots,t_s]}$ and s is the maximum sequence length.

• **Answer Type Classifier.** TAT-DQA offers four different answer types, i.e., *Span*, *Spans*, *Counting*, *Arithmetic*. We adopt an Answer Type Classifier to predict the answer type of a question, which is essentially a multi-class classifier taking the SD graph representation h_{SD} as input. The probability of each answer type is computed as

$$\mathrm{P}_{\textsf{type}} = \mathrm{softmax}(\mathrm{FFN}(h_{\mathrm{SD}})). \tag{3}$$

FFN is a feed-forward network with two layers.

• **Span Classifier.** For the *Span* question, the answer is a sub-sequence of the input sequence, which is achieved by the Span Classifier. It takes the token representations obtained in Section 2.4 as the input and predicts the start and end indices of the sub-sequence. Formally, the probability distribution of the start position over the sequence is obtained by

$$\mathrm{P}_{\textsf{start}} = \mathrm{softmax}(\mathrm{FFN}(h^{'}_{[t_1,t_2,\ldots,t_s]})) \tag{4}$$

where FFN is a feed-forward network with two layers. Similarly, we can obtain the probability of the end position $\mathrm{P}_{\textsf{end}}$.

• **Token Classifier.** For the *Spans* and *Counting* questions, a Token Classifier is employed to infer the final answer. In particular, for each valid token obtained in Section 2.4, Token Classifier assigns a `B`, `I` or `O` label and takes those tagged with `B` and `I` to generate the final answer. Formally, it takes in the updated representation $h^{'}_{token}$ of each valid token and computes the probability of the label as

$$\mathrm{P}_{\mathrm{token}} = \mathrm{softmax}(\mathrm{FFN}(h^{'}_{token})) \tag{5}$$

where FFN is a feed-forward network with two layers. After obtaining the tokens, the final answer for *Spans* and *Counting* questions is generated heuristically following MHST (Zhu et al., 2022).

• **Tree Generator.** For the *Arithmetic* question, a Tree Generator is adopted to generate an expression tree with the selected *Quantity* and *Date* nodes, which can be executed to infer the answer. Following MHST (Zhu et al., 2022), the Tree Generator is implemented with GTS (Xie and Sun, 2019), which generates expression trees in a goal-driven manner. To adapt GTS in our model, we make two major modifications. First, instead of feeding all the numbers and dates in the input into GTS, we only feed the selected most relevant *Quantity* and *Date* nodes, which significantly reduces the number of candidates for GTS to predict each leaf node and alleviates the difficulties. Second, when GTS predicts each node in the expression tree, we revise the generation of the context vector by attending to all the nodes in the SD graph instead of the tokens in the sequence, which can offer enhanced comprehensive semantic representations of the document.

Type	Model	EM	F_1
Human Expert Performance		84.10	90.80
Fine-tuned	NumNet+ V2	30.60	40.10
	TagOp	33.70	42.50
	MHST	41.50	50.70
LLMs	MAmmoTH (70B)	35.42	42.82
	WizardMath (70B)	36.44	41.55
	LLaMA 2-Chat (70B)	41.91	49.74
	ChatGPT	52.74	61.40
Ours	Doc2SoarGraph	(+6.49) **59.23**	(+6.21) **67.61**

Table 1: Performance of our model and baseline models on the test set of TAT-DQA.

The expression tree generated by the Tree Generator includes three kinds of nodes: the arithmetical operators V_{op} (i.e., +,-,*,/), the constant numbers V_{con} (i.e., 1,2,3, .., 100), and the quantity and date nodes V_{node} that are selected in Section 2.3. The target vocabulary for tree generation is denoted as $V = V_{op} \cup V_{con} \cup V_{node}$ and its length is denoted as L. Following the typical construction method of GTS (Xie and Sun, 2019), the expression tree is constructed starting from producing the topmost operator and then the left and right child nodes.

• **Scale Classifier.** Scale is vital for a numerical answer in TAT-DQA, including five possible values: *Thousand*, *Million*, *Billion*, *Percent* and *None*. A Scale Classifier is developed to predict the scale of the final answer. In particular, it takes as input the SD graph representation h_{SD} and computes the probability of each scale as

$$\mathrm{P}_{\mathrm{scale}} = \mathrm{softmax}(\mathrm{FFN}(h_{\mathrm{SD}})) \tag{6}$$

where FFN is a feed-forward network with two layers. After obtaining the scale, we generate the final answer by multiplying or concatenating the answer value with the scale following the practice in MHST (Zhu et al., 2022).

$$\begin{aligned}
\mathcal{L} =& \mathcal{L}_{node}+\mathcal{L}_{tree}+\mathcal{L}_{start}+\mathcal{L}_{end}+\mathcal{L}_{type}+\mathcal{L}_{token}+\\
&\mathcal{L}_{scale}\\
\mathcal{L}_{node} =& \frac{1}{|\mathrm{N}|}\sum_{\mathrm{n}\in\mathrm{N}}\mathrm{CE}(\mathrm{P}^{\mathrm{n}}_{\mathrm{node}}, \mathrm{g}^{\mathrm{n}}_{\mathrm{node}})\\
\mathcal{L}_{tree} =& \frac{1}{|\mathrm{S}|}\sum_{\mathrm{s}\in\mathrm{S}}\mathrm{CE}(\mathrm{P}(v^s|v^1, ..., v^{s-1}, Q, G), \mathrm{g}^s_v)\\
\mathcal{L}_{start} =& \mathrm{CE}(\mathrm{P}_{start}, \mathrm{g}_{start})\\
\mathcal{L}_{end} =& \mathrm{CE}(\mathrm{P}_{end}, \mathrm{g}_{end})\\
\mathcal{L}_{type} =& \mathrm{CE}(\mathrm{P}_{type}, \mathrm{g}_{type})\\
\mathcal{L}_{token} =& \frac{1}{|\mathrm{T}|}\sum_{\mathrm{t}\in\mathrm{T}}\mathrm{CE}(\mathrm{P}^{\mathrm{t}}_{\mathrm{token}}, \mathrm{g}^{\mathrm{t}}_{\mathrm{token}})\\
\mathcal{L}_{scale} =& \mathrm{CE}(\mathrm{P}_{scale}, \mathrm{g}_{scale}).
\end{aligned} \tag{7}$$

2.5. Training

To optimize the proposed model, the objective is to minimize the sum of the losses of all classification tasks. Formally, the overall loss for each sample can be computed as

Here N is a set of nodes; g^n_{node} is the ground-truth label if the node n is selected; $\mathrm{CE}(\cdot)$ refers to the cross-entropy loss; S is the number of decoding steps during the expression tree generation; g^s_v is the ground-truth node in the step s; g_{start} and g_{end} are the ground-truth starting and ending positions of the span answer; g_{type} is the ground-truth answer type; T refers to all the valid tokens after applying the token masking in Section 2.4; g^t_{token} is the ground-truth label of the token t; g_{scale} is the ground-truth scale value. When the nodes in the ground-truths in Node Selection are not selected, we will add them manually in order to better train the tree-based decoder when training.

3. Experiments

We conduct extensive experiments to validate the effectiveness of our proposed model and present comprehensive analyses.

3.1. Experiment Settings

• **Dataset.** We conduct all experiments on TAT-DQA (Zhu et al., 2022) built with visually-rich table-text documents in finance. It contains $16,558$ QA pairs on $2,758$ documents where each document contains at least one table. These documents are split into training, development and test sets with a ratio of $8:1:1$, and all the questions of a specific document belong to only one of the splits. Over 50% of questions require discrete reasoning to generate answers while the answers to others can be extracted directly from the documents.

• **Baselines.** We select two kinds of baselines: fine-tuned models and large language models (LLMs). **Fine-tuned models** are trained over TAT-DQA dataset, including: 1) NumNet+ V2 (Ran et al., 2019) is a text QA model with impressive capability of discrete reasoning over textual data. It constructs a numerically-aware graph neural network, which takes all numbers in the given question and document as nodes and builds edges via numerical comparison, and then performs discrete reasoning over the graph. 2) TagOp (Zhu et al., 2021) is a table-text QA model which first applies sequence tagging on each token to extract question-relevant ones and then applies a set of pre-defined aggregation operators (e.g. addition, counting) over extracted tokens. 3) MHST (Zhu et al., 2022) is a multi-modal QA model which employs LayoutLMv2 (Xu et al., 2021) as the encoder to take the question and document as input, extracts supporting evidence using sequence tagging, and applies a tree-based decoder (Xie and Sun, 2019) to generate an expression tree with the evidence. **LLMs** are tested directly in a zero-shot manner, including two general LLMs, LaMA 2-Chat (70B) (Touvron et al., 2023) and ChatGPT (Brown et al., 2020), and two LLMs specialized in math word problems (MWP), MAmmoTH (70B) (Yue et al., 2023) and WizardMath (70B) (Luo et al., 2023).

• **Evaluation Metrics.** Following (Zhu et al., 2022), Exact Match (EM) and numeracy-focused (macro-averaged) F_1 are used to measure the performance of all models, taking into account the scale of the answer. Both metrics are in the range of [0%, 100%], where a higher value indicates better performance.

• **Implementation Details.** We implement our model in PyTorch and train it on one NVIDIA DGX-1 with eight V100 GPUs. We adopt LayoutLMv2$_{large}$ as the encoder. We use Adam optimizer with learning rate $5e-4$ and warmup over the first 6% steps to train. The maximum number of epochs is set to 50 and the maximum sequence length 512. The batch size is set to 8 and the number of gradient accumulation steps is 8. The dropout probabilities for GCNs and GTS are 0.6 and 0.5 respectively while 0.1 for others. We set 12 as the maximum number of selected nodes in node selection. Beam search is applied during inference to select the best expression tree and the beam size is 5.

Given that the tabular and textual data in each document are typically lengthy, it is impractical to incorporate additional in-context examples owing to the input length constraints of the LLMs, thus we test all LLMs in a zero-shot setting. We utilize the latest ChatGPT[1] (Brown et al., 2020) APIs[2]. The parameters `temperature`, `top_p` and `max_tokens` are set with 0, 1.0 and $1,000$, and other parameters as default. We obtain the official trained checkpoints of LLaMA 2-Chat (Touvron et al., 2023), MAmmoTH (Yue et al., 2023) and WizardMath (Luo et al., 2023) from Hugginface[3]. The model inference is done on one NVIDIA DGX-A100 with eight A100 GPUs. The parameters `num_beam` and `do_sample` are 1 and false respectively.

3.2. Main Results

We first compare our Doc2SoarGraph model with all baseline models. The experimental results are shown in Table 1. We can observe that: 1) Our Doc2SoarGraph model significantly outperforms all baseline models. In particular, our model reaches 59.23% and 67.61% on the test set in terms of Exact Match and F_1 metrics respectively, i.e., an increase of 17.73 and 16.91 points over MHST (Zhu et al., 2022), and 6.49 and 6.21 points over ChatGPT. These results well demonstrate the great effectiveness of our model. 2) The LLMs specialized in mathematical reasoning, i.e. WizardMath (Luo et al., 2023) and MAmmoTH (Yue et al., 2023), still largely underperform our Doc2SoarGraph model, indicating that current numerically-enhanced LLMs still struggle in discrete reasoning over tabular and textual QA. 3) The best fine-tuned model MHST achieves comparable performance to the outstanding LLaMA-2 Chat with much smaller size, and Doc2SoarGraph largely outperforms the powerful ChatGPT. This shows that current general LLMs still struggle with table-text document QA, and fine-tuning on the dataset is still a promising approach.

[1]GPT3.5-Turbo (Sep 2023)

[2]https://platform.openai.com/.

[3]https://huggingface.co/models

Model	EM		F_1	
	MHST	Ours	MHST	Ours
Span	41.10	**50.00**	58.30	**62.88**
Spans	25.70	**41.43**	43.30	**71.19**
Counting	**43.20**	40.00	**43.20**	40.00
Arithmetic	42.70	**73.96**	42.70	**73.96**

Table 2: Performance comparison of our model and MHST for different answer types on TAT-DQA test set. Best results are marked in bold.

3.3. In-depth Analysis of Our Model

• **Analysis on Evidence Extraction.** Generally, for discrete reasoning over table-text documents, the model first extracts supporting evidence and then reasons over it.

Here we verify whether the evidence extraction power is indeed enhanced with our model. We compute the average recall, precision and F_1 score of the extracted evidence with our method and MHST on the dev set. For fairness, we only use the `Arithmetic` questions that only depend on quantity and date nodes. Given one question, assume the number of quantities/dates its answer actually requires is n, the number of predicted quantities/dates by the model is m and the number of correct quantities/dates in prediction is c. The recall and precision are computed with c/n and c/m respectively. Then we can compute the F_1 with the precision and recall. After getting the metrics of each question, we further obtain the average recall, precision and F_1.. The results are shown in Figure 3. We can see that our model demonstrates significant improvements over the MHST. Specifically, our method has an increase of 23.76 and 24.43 points in average precision and average recall compared with MHST, significantly improving the evidence extraction.

Model	EM (↑)	F_1 (↑)
MHST	41.50 (-)	50.70 (-)
+ Node Initialization	54.30 (12.80)	61.59 (10.89)
+ Doc Transformation	56.58 (2.28)	64.06 (2.47)
+ Hierarchical Graphs	58.80 (2.22)	66.60 (2.54)
+ Token Masking (Full)	**59.23** (0.43)	**67.61** (1.06)

Table 3: Analysis on effects of the components in Doc2SoarGraph on test set. Best results are marked in bold.

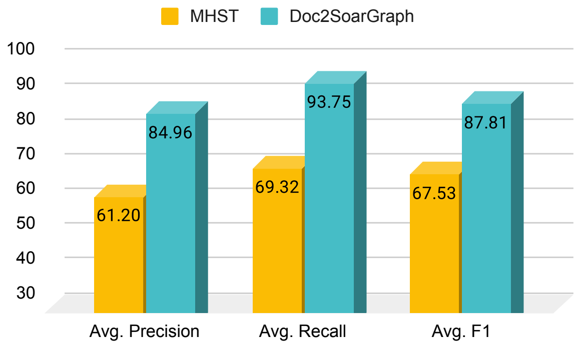

Figure 3: Comparison of evidence extraction power of our Doc2SoarGraph and MHST on *Arithmetic* questions on dev set.

• **Analysis on Answer Types.** TAT-DQA provides four different answer types, and here we analyze the performance of our model on each answer type. The results are summarized in Table 2. Compared with MHST, our model gains the largest increase (i.e., 31.26% in EM) on `Arithmetic` questions, demonstrating impressive discrete reasoning capability. This enhancement is possibly due to the effective modeling of the differences and correlations among the quantities, dates and blocks from the documents. For `Spans` and `Counting` questions, they share almost all techniques in the proposed model. Comparably, the model gains a 15.72% increase on `Spans` questions but has a 3.2% decrease on `Counting` (i.e., failing one more case). This is probably due to the data bias on `Counting` questions because the number of `Counting` questions ($<2.0\%$) is much less than `Spans` ($>12.0\%$) on test set. The model obtains an increase of 8.9% in EM on `Span` questions, indicating our design also benefits answer extraction from the document.

• **Analysis on Single- and Multi-page Documents.** We analyze the performance differences of three models on single-page documents and multi-page documents, i.e. MHST, ChatGPT and our Doc2SoarGraph model. See Figure 4 for the comparison results. We make following observations. 1) All three models perform better on single-page documents than on multi-page ones, implying that it is more challenging to understand multi-page documents than single-page ones. 2) Our model outperforms MHST and ChatGPT with large margins for understanding single-page documents, further indicating the effectiveness of our model. 3) For multi-page document understanding, the performance of our model is better than MHST but worse than ChatGPT. The inferiority of our model to ChatGPT is mostly possibly due to the much shorter input allowed by our model.

Model	Dev		Test	
	EM	F_1	EM	F_1
Full Graphs	**57.97**	**65.38**	**59.23**	**67.61**
- QC Graph	56.69	65.18	57.73	66.89
- DC Graph	56.14	64.83	57.73	66.82
- TR Graph	55.23	63.57	55.86	65.09
- SD Graph	56.27	64.77	56.95	66.54

Table 4: Ablation study of the hierarchical graphs in our model on test set.

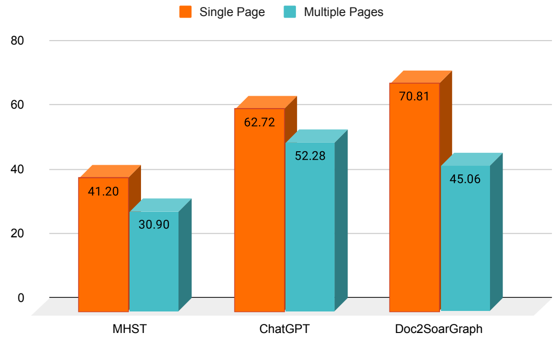

Figure 4: Performance comparison in F_1 score on one- and multi-page documents on test set.

• **Analysis on Model Components.** Our Doc2SoarGraph contains four steps, i.e., Multi-page Document Transformation, Node Initialization according to various semantic elements, Node Selection via hierarchical graphs, and Answer Generation powered by token masking. Here we investigate the contributions of each component to its final performance. Compared to MHST, our Doc2SoarGraph is equipped with the four components, and it is found that all added components can benefit model performance. Furthermore, we find node initialization makes surprisingly greater contributions to the model, indicating the importance of modeling the differences and correlations among various elements in table-text documents.

Also, we develop our model with hierarchical graphs, i.e. the QC graph, DC graph, TR graph and SD graph. To test the necessity of each graph, we remove each of them to see the performance changes. The results are summarized in Table 4. Performance drop can be observed as we remove each graph, indicating that each graph contributes to the good performance of our model.

• **Error Analysis.** We randomly sample 100 error instances of our method on dev set and analyze the reasons. We find the errors occur to all six

Module	Error	%
Span Classifier (SPC)	Offset Error	21%
	No Overlap	11%
Node Classifier (NC)	Missing Nodes	24%
Token Classifier (TC)	Missing Tokens	11%
	Redundant Tokens	8%
Tree Generator (TG)	Wrong Expression	12%
	Wrong Sign	4%
Scale Classifier (SC)	Wrong Scale	6%
Answer Type Classifier (ATC)	Wrong Answer Type	3%

Table 5: Statistics of errors in each module.

modules (Col. 1 in Table 5), i.e. Span Classifier (SPC), Token Classifier (TC), Node Classifier (NC), Tree Generator (TG), Scale Classifier (SC) and Answer Type Classifier (ATC), listed in a descending order of error percentage. These errors are classified into nine categories (Col. 2 in Table 5). We can see, 1) 32% errors are caused by SPC module predicting inaccurate predictions of starting and ending positions for Span questions, i.e., 21% predictions overlapping but not exactly matching ground truth, and 11% predictions having zero overlap with ground truth; 2) 24% errors are caused by NC module failing to select the relevant nodes; 3) 19% errors are due to TC module predicting less or more tokens than it needs to derive the answer; 4) 16% errors are caused by TG module generating a wrong expression tree, among which 4% are wrong number signs (i.e., positive/negative) and 12% are other wrong expressions; 5) 6% and 3% errors are caused by SC module and ATC module predicting wrong scale and answer types.

4. Related Work

4.1. Document VQA

Document VQA aims to answer a question in natural language based on a visually-rich document (Cui et al., 2021; Mathew et al., 2020; Tanaka et al., 2021; Zhu et al., 2022). Compared to typical VQA, the documents in this challenges like DocVQA (Mathew et al., 2020), VisualMRC (Tanaka et al., 2021) and TAT-DQA (Zhu et al., 2022) usually contain rich textual information that plays a key role in addressing the challenge. It is mostly tackled by pre-trained language models, e.g. LAMBERT (Garncarek et al., 2020), StructuralLM (Li et al., 2021a) which exploit both textual and layout information of the documents. Some works develop multi-modal language models that incorporate visual information into the model, e.g., LayoutLMv2 (Xu et al., 2021) and DocFormer (Appalaraju et al., 2021). Additionally, some DocVQA models are developed by fine-tuning pre-trained language models, e.g. TILT (Powalski et al., 2021) and MHST (Zhu et al., 2022). Recently, large-scale language models like ChatGPT (Brown et al., 2020) have achieved impressive results across a range of natural language processing (NLP) tasks (Zhao et al., 2023). In this work, we develop Doc2SoarGraph to comprehend visually-rich table-text documents by extending SoarGraph (Zhu et al., 2023), achieving comparable performance with the very large-scale language models like ChatGPT (Brown et al., 2020).

4.2. Discrete Reasoning

Discrete reasoning has been explored in many NLP tasks since 1960s (Feigenbaum et al., 1963; Dua et al., 2019). Recent works focus on a hybrid of annotated (semi-)structured table and a list of associated paragraphs (Chen et al., 2021; Zhao et al., 2022; Zhu et al., 2021), retrieving or extracting evidences from given table and paragraphs and then reasoning over evidences to generate the answer (Lei et al., 2022; Zhou et al., 2022; Li et al., 2022a; Nararatwong et al., 2022; Yarullin and Isaev, 2023; Zhu et al., 2021). Most recently, a document VQA dataset TAT-DQA (Zhu et al., 2022) is released, which triggers the increasing interest in discrete reasoning over real-world complex documents with both tables and text. To tackle this challenging task, Zhu et al. (2022) proposed the MHST model, which extracts relevant tokens from the document using sequence tagging and applies heuristic or "seq2tree" method to generate the answer according to the answer type. We also address the TAT-DQA challenge but with a more powerful model.

4.3. Graph-based Document Representation

Early works use grid-based methods to represent visually-rich documents, such as representing each document page as a grid of characters (Katti et al., 2018) or a grid of contextualized word piece embedding vectors (Denk and Reisswig, 2019). Later, many works (Riba et al., 2019; Hwang et al., 2021; Wei et al., 2020; Yu et al., 2020; Cheng et al., 2020) represent documents with more powerful graphs to facilitate information extraction from visually-rich documents. For example, (Riba et al., 2019) adopts a GNN-based model to extract structured tables from invoices; (Hwang et al., 2021) constructs a directed graph to model the spatial dependency among text tokens in the documents. In this work, we represent the question and document by building hierarchical graphs with different semantic elements in the document (i.e., quantities, dates, and document blocks).

5. Conclusion

In this work, we propose a novel Doc2SoarGraph model with strong discrete reasoning capabilities to tackle QA challenge over visually-rich table-text documents in the form of TAT-DQA, which models the differences and correlations of various elements (i.e., quantities, dates, question, and document blocks) in the input with hierarchical graphs. We experimentally validate that our model can beat previous state-of-the-art by large margins. In the future, we would like to explore more advanced methods to handle the challenging multi-page documents, even very long ones like financial statements with over 100 pages.

Limitations

Despite the impressive performance on TAT-DQA (Zhu et al., 2022), our model still has much room for future improvement, as shown in error analysis in Section 3.3. Also, the Document Transformation technique that we have developed for preprocessing multi-page documents is simple, and more effective methods are desired. For example, our model may not be applicable to the documents with a large number of pages (e.g., >50 pages). In addition, our model is designed for the documents that contain different kinds of elements, such as numerical values and dates. This means it may have limited advantages over those with unique elements like pure textual documents.

Furthermore, our model has two major limitations on its discrete reasoning capabilities. First, the model is trained on TAT-DQA that is constructed with financial statements in the finance domain, which may result in limited applicability in other domains. Second, the types of discrete reasoning supported are restricted to those in the benchmark, which currently includes operations such as addition, subtraction, multiplication, division, counting, comparison, sorting, and combinations.

Ethics Statement

In this work, we present a new model to boost the performance of discrete reasoning over visually-rich table-text documents. Our model is developed on open-source tools and datasets to assist human-being in process . Thus, we do not anticipate any negative social impacts.

Acknowledgements

The authors gratefully thank all the anonymous reviewers for their positive feedback. This research is supported by the NExT Research Center, Singapore.

6. Bibliographical References

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. 2019. Giving BERT a calculator: Finding operations and arguments with reading comprehension. In *EMNLP-IJCNLP*, pages 5947–5952. ACL.

Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R. Manmatha. 2021. Docformer: End-to-end transformer for document understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 993–1003.

Daniel G Bobrow. 1964. Natural language input for a computer problem solving system.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901.

Kunlong Chen, Weidi Xu, Xingyi Cheng, Zou Xiaochuan, Yuyu Zhang, Le Song, Taifeng Wang, Yuan Qi, and Wei Chu. 2020. Question directed graph attention network for numerical reasoning over text. In *EMNLP-IJCNLP*, pages 6759–6768. ACL.

Xingyu Chen, Zihan Zhao, Lu Chen, JiaBao Ji, Danyang Zhang, Ao Luo, Yuxuan Xiong, and Kai Yu. 2021. WebSRC: A dataset for web-based structural reading comprehension. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4173–4185. Association for Computational Linguistics.

Mengli Cheng, Minghui Qiu, Xing Shi, Jun Huang, and Wei Lin. 2020. One-shot text field labeling using attention and belief propagation for structure information extraction. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 340–348. Association for Computing Machinery.

Ting-Rui Chiang and Yun-Nung Chen. 2019. Semantically-aligned equation generation for solving and reasoning math word problems. 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 2656–2668. Association for Computational Linguistics.

Lei Cui, Yiheng Xu, Tengchao Lv, and Furu Wei. 2021. Document AI: benchmarks, models and applications. *CoRR*, abs/2111.08609.

Timo I. Denk and Christian Reisswig. 2019. {BERT}grid: Contextualized embedding for 2d document representation and understanding. In *Workshop on Document Intelligence at NeurIPS 2019*.

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

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 *Proc. of NAACL*.

Edward A Feigenbaum, Julian Feldman, et al. 1963. *Computers and thought*. New York McGraw-Hill.

Lukasz Garncarek, Rafal Powalski, Tomasz Stanislawek, Bartosz Topolski, Piotr Halama, and Filip Gralinski. 2020. LAMBERT: layout-aware language modeling using BERT for information extraction. *CoRR*, abs/2002.08087.

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

Teakgyu Hong, DongHyun Kim, Mingi Ji, Wonseok Hwang, Daehyun Nam, and Sungrae Park. 2021. {BROS}: A pre-trained language model for understanding texts in document.

Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. 2019. A multi-type multi-span network for reading comprehension that requires discrete reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1596–1606. Association for Computational Linguistics.

Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. 2017. Learning fine-grained expressions to solve math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 805–814. ACL.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 4083–4091. Association for Computing Machinery.

Wonseok Hwang, Jinyeong Yim, Seunghyun Park, Sohee Yang, and Minjoon Seo. 2021. Spatial dependency parsing for semi-structured document information extraction. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 330–343. Association for Computational Linguistics.

Nengzheng Jin, Joanna Siebert, Dongfang Li, and Qingcai Chen. 2022. A survey on table question answering: Recent advances. In *China Conference on Knowledge Graph and Semantic Computing*, pages 174–186. Springer.

Anoop R Katti, Christian Reisswig, Cordula Guder, Sebastian Brarda, Steffen Bickel, Johannes Höhne, and Jean Baptiste Faddoul. 2018. Chargrid: Towards understanding 2D documents. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4459–4469. Association for Computational Linguistics.

Thomas N. Kipf and Max Welling. 2017a. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Thomas N. Kipf and Max Welling. 2017b. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 271–281. ACL.

Fangyu Lei, Shizhu He, Xiang Li, Jun Zhao, and Kang Liu. 2022. Answering numerical reasoning questions in table-text hybrid contents with graph-based encoder and tree-based decoder.

In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1379–1390, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Chenliang Li, Bin Bi, Ming Yan, Wei Wang, Songfang Huang, Fei Huang, and Luo Si. 2021a. StructuralLM: Structural pre-training for form understanding. 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 6309–6318. Association for Computational Linguistics.

Chenying Li, Wenbo Ye, and Yilun Zhao. 2022a. Finmath: Injecting a tree-structured solver for question answering over financial reports. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6147–6152.

Moxin Li, Fuli Feng, Hanwang Zhang, Xiangnan He, Fengbin Zhu, and Tat-Seng Chua. 2022b. 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. Association for Computational Linguistics.

Peizhao Li, Jiuxiang Gu, Jason Kuen, Vlad I. Morariu, Handong Zhao, Rajiv Jain, Varun Manjunatha, and Hongfu Liu. 2021b. Selfdoc: Self-supervised document representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5652–5660.

Weihong Lin, Qifang Gao, Lei Sun, Zhuoyao Zhong, Kai Hu, Qin Ren, and Qiang Huo. 2021. Vibertgrid: A jointly trained multi-modal 2d document representation for key information extraction from documents. In *Document Analysis and Recognition – ICDAR 2021*, pages 548–563. Springer International Publishing.

Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier, and Julian Martin Eisenschlos. 2022. Matcha: Enhancing visual language pretraining with math reasoning and chart derendering. *arXiv preprint arXiv:2212.09662*.

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. Association for Computational Linguistics.

Xiaojing Liu, Feiyu Gao, Qiong Zhang, and Huasha Zhao. 2019b. Graph convolution for multimodal information extraction from visually rich documents. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 32–39. Association for Computational Linguistics.

Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2022. A survey of deep learning for mathematical reasoning. *arXiv preprint arXiv:2212.10535*.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct.

Minesh Mathew, Viraj Bagal, Rubèn Pérez Tito, Dimosthenis Karatzas, Ernest Valveny, and C. V Jawahar. 2021. Infographicvqa.

Minesh Mathew, Dimosthenis Karatzas, R. Manmatha, and C. V. Jawahar. 2020. Docvqa: A dataset for VQA on document images. *CoRR*, abs/2007.00398.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2144–2153. ACL.

Rungsiman Nararatwong, Natthawut Kertkeidkachorn, and Ryutaro Ichise. 2022. 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 (Volume 1: Long Papers)*, pages 991–1000. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1470–1480. ACL.

Qiming Peng, Yinxu Pan, Wenjin Wang, Bin Luo, Zhenyu Zhang, Zhengjie Huang, Teng Hu, Weichong Yin, Yongfeng Chen, Yin Zhang, et al.

2022. Ernie-layout: Layout knowledge enhanced pre-training for visually-rich document understanding. *arXiv preprint arXiv:2210.06155*.

Rafal Powalski, Lukasz Borchmann, Dawid Jurkiewicz, Tomasz Dwojak, Michal Pietruszka, and Gabriela Palka. 2021. Going full-tilt boogie on document understanding with text-image-layout transformer. *CoRR*, abs/2102.09550.

Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. NumNet: Machine reading comprehension with numerical reasoning. In *EMNLP-IJCNLP*, pages 2474–2484.

Pau Riba, Anjan Dutta, Lutz Goldmann, Alicia Fornés, Oriol Ramos, and Josep Lladós. 2019. Table detection in invoice documents by graph neural networks. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 122–127.

Ryota Tanaka, Kyosuke Nishida, and Sen Yoshida. 2021. Visualmrc: Machine reading comprehension on document images. In *AAAI*.

Rubèn Tito, Dimosthenis Karatzas, and Ernest Valveny. 2022. Hierarchical multimodal transformers for multi-page docvqa.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Dingzirui Wang, Longxu Dou, and Wanxiang Che. 2022a. A survey on table-and-text hybridqa: Concepts, methods, challenges and future directions. *arXiv preprint arXiv:2212.13465*.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018. Translating a math word problem to a expression tree. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1064–1069. Association for Computational Linguistics.

Wenjin Wang, Zhengjie Huang, Bin Luo, Qianglong Chen, Qiming Peng, Yinxu Pan, Weichong Yin, Shikun Feng, Yu Sun, Dianhai Yu, and Yin Zhang. 2022b. Mmlayout: Multi-grained multimodal transformer for document understanding. MM '22, page 4877–4886. Association for Computing Machinery.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854. Association for Computational Linguistics.

Mengxi Wei, YIfan He, and Qiong Zhang. 2020. Robust layout-aware ie for visually rich documents with pre-trained language models. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 2367–2376. Association for Computing Machinery.

Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *IJCAI*, pages 5299–5305.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. 2021. LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. 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 2579–2591. Association for Computational Linguistics.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 1192–1200. Association for Computing Machinery.

Ramil Yarullin and Sergei Isaev. 2023. Numerical embeddings for reasoning over text and tables.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *ACL*, pages 8413–8426. ACL.

Wenwen Yu, Ning Lu, Xianbiao Qi, Ping Gong, and Rong Xiao. 2020. PICK: Processing key information extraction from documents using improved graph learning-convolutional networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning.

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. Association for Computational Linguistics.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models.

Yongwei Zhou, Junwei Bao, Chaoqun Duan, Youzheng Wu, Xiaodong He, and Tiejun Zhao. 2022. Unirpg: Unified discrete reasoning over table and text as program generation. *arXiv preprint arXiv:2210.08249*.

Fengbin Zhu, Wenqiang Lei, Fuli Feng, Chao Wang, Haozhou Zhang, and Tat-Seng Chua. 2022. Towards complex document understanding by discrete reasoning. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4857–4866.

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. Association for Computational Linguistics.

Fengbin Zhu, Moxin Li, Junbin Xiao, Fuli Feng, Chao Wang, and Tat Seng Chua. 2023. Soargraph: Numerical reasoning over financial table-text data via semantic-oriented hierarchical graphs. In *Companion Proceedings of the ACM Web Conference 2023*, page 1236–1244. Association for Computing Machinery.

7. Language Resource References

Chen, Wenhu and Zha, Hanwen and Chen, Zhiyu and Xiong, Wenhan and Wang, Hong and Wang, William Yang. 2020. *HybridQA: A Dataset of Multi-Hop Question Answering over Tabular and Textual Data*. Association for Computational Linguistics.

Chen, Zhiyu and Chen, Wenhu and Smiley, Charese and Shah, Sameena and Borova, Iana and Langdon, Dylan and Moussa, Reema and Beane, Matt and Huang, Ting-Hao and Routledge, Bryan and Wang, William Yang. 2021. *FinQA: A Dataset of Numerical Reasoning over Financial Data*. Association for Computational Linguistics.

Zhao, Yilun and Li, Yunxiang and Li, Chenying and Zhang, Rui. 2022. *MultiHiertt: Numerical Reasoning over Multi Hierarchical Tabular and Textual Data*. Association for Computational Linguistics.

Zhu, Fengbin and Lei, Wenqiang and Feng, Fuli and Wang, Chao and Zhang, Haozhou and Chua, Tat-Seng. 2022. *Towards complex document understanding by discrete reasoning*.

Zhu, Fengbin and Lei, Wenqiang and Huang, Youcheng and Wang, Chao and Zhang, Shuo and Lv, Jiancheng and Feng, Fuli and Chua, Tat-Seng. 2021. *TAT-QA: A Question Answering Benchmark on a Hybrid of Tabular and Textual Content in Finance*. Association for Computational Linguistics.