Enhancing Financial Table and Text Question Answering with Tabular Graph and Numerical Reasoning

Rungsiman Nararatwong[1], Natthawut Kertkeidkachorn[2], Ryutaro Ichise[3,1]
[1]National Institute of Advanced Industrial Science and Technology
[2]Japan Advanced Institute of Science and Technology
[3]Tokyo Institute of Technology
r.nararatwong@aist.go.jp, natt@jaist.ac.jp
ichise@iee.e.titech.ac.jp

Abstract

Typical financial documents consist of tables, texts, and numbers. Given sufficient training data, large language models (LM) can learn the tabular structures and perform numerical reasoning well in question answering (QA). However, their performances fall significantly when data and computational resources are limited. This study improves this performance drop by infusing explicit tabular structures through a graph neural network (GNN). We proposed a model developed from the baseline of a financial QA dataset named TAT-QA. The baseline model, TagOp, consists of answer span (evidence) extraction and numerical reasoning modules. As our main contributions, we introduced two components to the model: a GNN-based evidence extraction module for tables and an improved numerical reasoning module. The latter provides a solution to TagOp's arithmetic calculation problem specific to operations requiring number ordering, such as subtraction and division, which account for a large portion of numerical reasoning. Our evaluation shows that the graph module has the advantage in low-resource settings, while the improved numerical reasoning significantly outperforms the baseline model.

1 Introduction

Working with tables and numerical reasoning is essential to understanding financial documents. However, off-the-shelf pre-trained LMs generally do not understand tables and numbers. Previous QA studies on tabular data either add specialized components to LMs then finetune or modify the LMs' architecture and pre-train with tables. The issue with these approaches is that they are not flexible to hybrid table-text data. Yet, and crucially, the model must be able to handle both data types, or it will fail to capture all the information in the documents.

In 2021, (Zhu et al., 2021) introduced TAT-QA, a dataset with the abovementioned challenges. It is a collection of financial reports with questions, some requiring arithmetic operation – as part of numerical reasoning – on the evidence extracted from the table, text, or both. The authors also published a model named TagOp, an LM with multiple classification heads for table and text-based evidence extraction and numerical reasoning. The model combines table and text as an input, performs evidence extraction, then applies numerical operations if needed. Our experimentation with TagOp led to two proposed components presented in this study.

The first component stems from how TagOp handles tables. The model takes a flattened table – a sequential concatenation of table cells – as an input without additional tabular structure information. Given sufficiently large training data, the model can learn the structure well by itself. However, it appears to struggle to understand tables with fewer training samples. Thus, we explicitly introduced graph-based tabular structural information through GNN, aiming to help the model understand tables without needing extensive labeling.

The second part of this study involves a specific classification head that determines the number order required for certain arithmetic operations, including subtraction and division. TagOp has this classifier, but its algorithm unintentionally introduces noise (irrelevant or invalid samples) that deters the model from recognizing meaningful patterns to generalize. Our solution selects relevant data, eliminates the noise, and includes an algorithm that handles this operation during training and inference. These operations account for a large part of numerical reasoning, emphasizing the importance of this problem.

Both methods have proved effective in different settings, thus designating our main contributions. The tabular graph module improves the model's understanding of tables in low-resource settings (small model and sample sizes). The number order classification component helps the model generalize, resulting in better performance. This work

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
November 20–23, 2022.

benefits QA and other relevant tasks that involve tables, especially in combination with texts, particularly in low-resource settings. It also advances QA models' numerical reasoning ability through a better training approach.

2 Background

This financial QA study proposes two methods to the baseline model of the TAT-QA dataset (TagOp). In this section, we will explain both the dataset and the baseline model, together with relevant works in tabular QA and numerical reasoning. Afterward, in the next section, we will revisit TagOp to present the problems and our approaches to improve the model.

2.1 Hybrid dataset

While there have been QA datasets focusing on texts (Rajpurkar et al., 2016), tables (Iyyer et al., 2017), and a combination of both (Chen et al., 2020), TAT-QA took a step closer to an actual application in the financial domain. Not only that it is a large-scale collection of hybrid text and table data with QA, but it also requires numerical reasoning. These properties make it even more challenging than the other datasets, and the authors showed that existing methods still left a large gap for improvement.

The dataset contains 16,552 questions with 2,757 hybrid contexts from 182 financial reports, splitting into 80% training set, 10% development set, and 10% test set. Each context includes one table and at least two associated paragraphs. Many questions require numerical reasoning, such as addition, subtraction, multiplication, division, counting, and comparison. The annotators created question-answer pairs from the contexts, together with derivations, which explain the steps taken to derive the answers.

2.2 TAT-QA's baseline

The authors of the TAT-QA dataset published a baseline model named TagOp along with the dataset. TagOp is an LM (they used RoBERTa; (Liu et al., 2019)) with multiple classification heads fine-tuned to extract evidence and determine the reasoning operations. The model first locates supporting evidence from table cells or text spans using the Inside-Outside (IO) sequence tagging approach (Ramshaw and Marcus, 1995). The input concatenates a question, flattened table by row (Herzig et al., 2020), and associated paragraphs sorted by TF-IDF scores. The tagging classifier is a two-layer feed-forward network (FNN) with GELU (Hendrycks and Gimpel, 2016) activation function. Given a sub-token t's representation h_t, the classifier outputs:

$$\mathbf{p}_{\mathrm{t}}^{\mathrm{tag}} = \mathrm{softmax}(\mathrm{FFN}(h_t)) \quad (1)$$

Once the model has identified the evidence, it determines the operation and calculates the answer if needed. This reasoning step involves three classifiers for the operator, number order, and scale; all are two-layer FFN with GELU activation function. There are ten operators in TagOp: *span-in-text*, *cell-in-table*, *spans*, *sum*, *count*, *average*, *multiplication*, *division*, *difference*, and *change ratio*. Three of the ten operators are number-order sensitive, including *division*, *difference*, and *change ratio*. Since TAT-QA also requires a scale of the answer, TagOp's scale classifier outputs *thousand*, *million*, *billion*, *percent*, or no scale. The three classifiers take different inputs as follows:

$$\mathbf{p}^{\mathrm{op}} = \mathrm{softmax}(\mathrm{FFN}(h_{cls})) \quad (2)$$

$$\mathbf{p}^{\mathrm{order}} = \mathrm{softmax}(\mathrm{FFN}(\mathrm{avg}(h_{t1}, h_{t2}))) \quad (3)$$

$$\mathbf{p}^{\mathrm{scale}} = \mathrm{softmax}(\mathrm{FFN}([h_{cls}; h_{tab}; h_p])) \quad (4)$$

h_{cls} is the representation of a sentence-level classification token. h_{t1} and h_{t2} are the output representations of the top two subtokens by the evidence extraction scores. h_{tab} and h_p averages table and paragraphs' subtoken respectively.

2.3 Related works

We compared our model to several baselines, including those reported in the TagOp study. The first baseline is BERT-RC (Devlin et al., 2019), or BERT for reading comprehension (RC). Another RC model is NumNet+ V2 (Ran et al., 2019), which performs well on DROP, a QA dataset with numerical reasoning on textual data (Dua et al., 2019). While these two models work well on texts, TaPas is an LM tailored to tabular input (Herzig et al., 2020), pre-trained on large-scale tables and associated texts from Wikipedia. The model in comparison is TaPas for WikiTableQuestion (WTQ). HyBrider (Chen et al., 2020), on the other hand, can handle both tables and texts without the limitations the previously mentioned models have.

In addition to the abovementioned baselines, we considered two post-TagOp models named KIQA

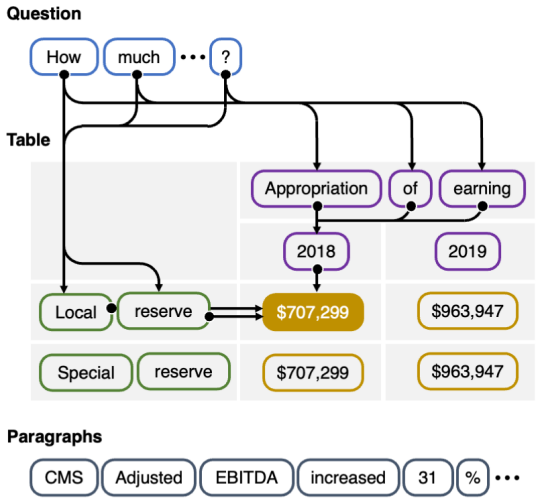

Figure 1: An example of a tabular graph linking tokens in the question through the table's row and column heads to a particular cell. The entire graph consists of these connections for all cells in the table. There are no links to tokens in the paragraphs.

and FinMath. KIQA (Nararatwong et al., 2022) is an entity retrieval model that replaces RoBERTa with LUKE (Yamada et al., 2020) to infuse external knowledge extracted by GENRE (Cao et al., 2021) into the LM. FinMath enhances the numerical reasoning capability by injecting a numerical expression tree into the model for the multi-step calculation (Li et al., 2022). Both models outperform TagOp on the TAT-QA dataset.

3 Methodology

We proposed two approaches developed from the TagOp model, each tackling a different problem but combined as a single complete model. The first three subsections will elaborate on the issues and methods; the last one explains the challenges of integrating the new components into the model and our final, most effective approach to achieve this objective.

3.1 Graph-based tabular evidence extraction

The issue with a typical LM is their lack of ability to understand tables. Directly including tables in the pre-training stage is expensive and inflexible to future changes to the underlying model architecture. The alternative is finetuning, which appears to work well given enough data, but that strategy alone could also come at a high cost. Therefore, we hypothesized that if the problem was because the model needs more to learn, we could help it learn by injecting our knowledge.

GNN was our choice due to its compatibility with tables: It can model the cells' relations to their respective row/column headers, is flexible to various structures, and does not require pre-training. Our simple heuristic algorithm can locate column headers with sufficiently high accuracy, and in most tables, only the first column is the row header. Specifically, the algorithm makes use of patterns we observed from the tables. It checks the table from top to bottom to identify the first row that meets particular criteria, such as containing numbers or empty, as a non-header. Complete detail of the rules is available as the supplemental material and source code. We manually annotated the header rows for evaluation, and the algorithm achieved 99.1% accuracy. These two findings, although not perfect, give us enough information to build the tabular graphs. We used GraphSAGE (Hamilton et al., 2017), which computes node embeddings by sampling and aggregating features from a node's local neighborhood.

As shown in Figure 1, the tabular graph maps each cell to its row/column headers with directed edges connecting all tokens in the header cell to those in the target cell. We only link header cells at the bottom of the hierarchy to the target cells for complex tables with hierarchical column headers. Cells in column headers have links to all header cells at the higher level (row), regardless of whether or not they have any actual connection. This strategy relies on the GNN to determine relationships among columns instead of explicitly telling the model which header cells to merge since the dataset does not provide such information.

Cells in the first column connect differently from column headers and the rest of the table. Row headers can also have a hierarchy, and it is as challenging to identify their links since they are not always explicit. We again linked every cell row-by-row from top to bottom and let the GNN decide what the hierarchy looks like through message passing. Lastly, we created full token-level links from the question to all row/column header cells.

3.2 Number order problem

Before introducing our method for the number order classification, first, we will clarify the problem and why it is crucial. TagOp's operator classifier is remarkably accurate for TAT-QA's simple math problems. Still, subtraction and division (also in extension, the change-ratio operation) require the

operands to be in a specific order. The issue we found was that the number order classifier did not always get the correct operands to train. Instead, it learned from noisy inputs interfering with its generalization ability. To make this problem clear, we will first explain the original algorithm.

Once the LM outputs token representations, the evidence extraction classifier computes the final scores to choose the answer spans. The number order module ranks these scores and picks the top two words, including numbers, from the entire input sequence, which covers the question, flattened table, and paragraphs. The algorithm then selects the ranked inputs from samples with the operator predicted as subtraction, division, or change-ratio. Finally, the number order classifier determines whether or not it should reverse the operand order. At this point, the module calculates the loss before combining it with other losses.

There are two problems with this algorithm. First, the number order classifier should only get the representations of relevant operands during training; otherwise, it would simply be learning noises. Second, relying on the operator classifier's predictions means that some irrelevant samples could also interfere with the training process, adding to the noises already caused by the first problem. Thus, our algorithm aims to ensure that we train the model with all relevant samples and numbers and filter out those that are not.

3.3 Number order classification

Instead of ranking by the evidence extraction scores, we masked all the irrelevant tokens, leaving only the two operands. The model then classifies the numbers into two classes for the first and second positions. We can now compute the cross-entropy loss, which concludes the forward pass.

During inference, however, we cannot create masks from the labels. Instead, the algorithm produces them on the fly from the evidence extraction step. First, the preprocessing step identifies numbers in the input sequence. Once the evidence extraction module assigns prediction scores to all tokens, the intermediate algorithm chooses two numbers with the highest scores, i.e., most likely to be the answers. It then masks all subtokens that do not belong to the selected numbers before inputting the masked representations into the number order classifier, which outputs the order prediction.

While this approach relies on the operator classifier, it only does so during inference, which means it will not affect the training. The intuition is that if the evidence were wrong, the reasoning would not matter, but the model should now reason more reliably given the correct evidence.

Since our number order classification module uses number masks, we now classify every token into three classes: the first and second operand and non-operand. We, therefore, revised Equation 3 to:

$$\mathbf{p}^{\text{order}} = \text{softmax}(\text{FFN}(h_t)) \tag{5}$$

where h_t is a token representation. The algorithm chooses the numbers most likely belong to the first and second classes as the operands. If it chose both numbers and the first operand, the less likely number would become the second operand.

3.4 The complete model

Multi-task learning can have positive, negative, or no effect on the tasks involved (Fifty et al., 2021; Aghajanyan et al., 2021; Aribandi et al., 2022). As we integrated our modules into the existing architecture, we kept track of changes to the other classifiers. We found that using GNN's output for classification other than predicting evidence from the table can cause varying detrimental effects. This problem is likely because the graphs only map the tabular relationship. Passing the LM's output through another layer of neural network that does not serve any purpose other than handling a table can only add to the error. Therefore, as shown in Figure 2, the scale, operator, and text-based evidence classifiers remain the same; they process information passed directly from the LM.

The only change the GNN module affects is the number order module since it depends on the extracted evidence for classification. In this case, we used the token representations from the GNN module instead of directly from the LM. To sum up, we proposed two solutions to make the model more robust in low-resource settings and perform numerical reasoning better while maintaining minimal impact on the other classifiers.

4 Experiments

We developed four models for evaluation, one as a reimplementation of TagOp, and the other three for the methods proposed. This section will explain the changes we have made to TagOp that are not part of the proposed methods, including the preprocessing of the data and the prediction steps.

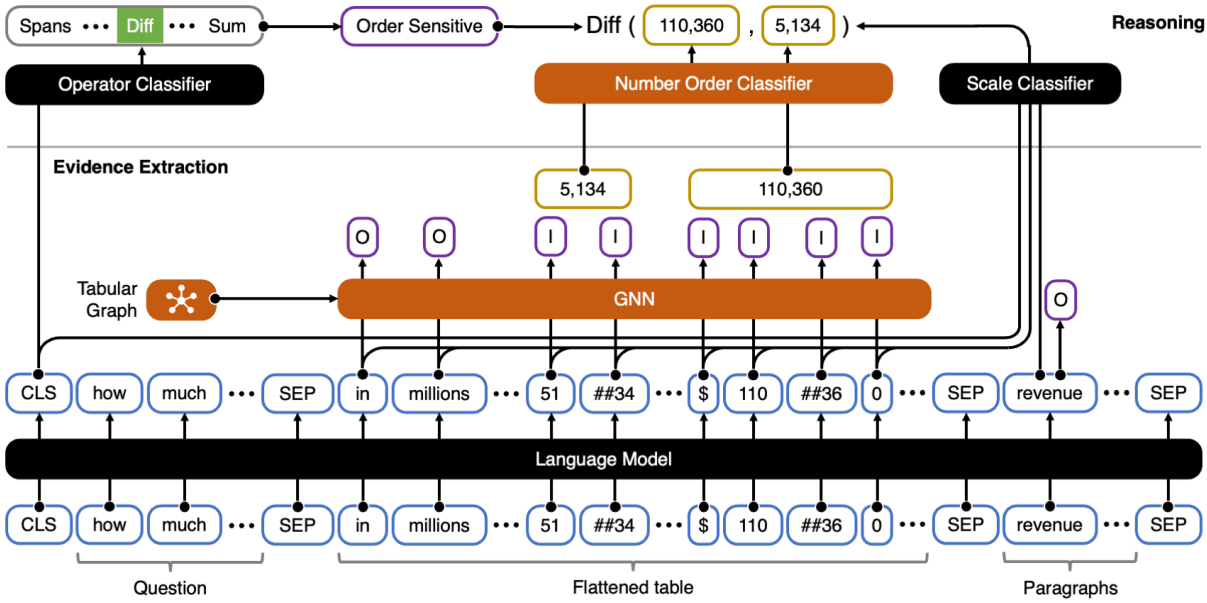

Figure 2: The proposed model develops from TagOp by adding the GNN module and introducing our number order classification modules, highlighted in orange. The tabular graph component automatically extracts a table structure and transforms it into a graph.

We used our reimplementation as the baseline for comparison to isolate the differences our modules cause and ensure a strictly controlled environment. The section will begin with the dataset and how we prepared it for low-resource settings, followed by model variation and evaluation metrics, three experiments we conducted, and the comparison with the baselines.

4.1 Dataset

The TAT-QA dataset has 16,552 questions extracted from 182 financial reports and split into 80% training, 10% development, and 10% test sets. Along with the answers are manually annotated derivations explaining the calculation, which we used to construct machine-readable labels following the baseline implementation. Consider the following question: "*What was the percentage change in the number of appliances in 2019 from 2018?*" The annotator labeled "*(680 - 774) / 774*" as the derivation, given that 680 and 774 are the numbers of appliances. Automatically determining the operator from this derivation is relatively straightforward.

However, we could not convert all derivations into tags since some do not constitute patterns suitable for automatic extraction (4.3% of the training and 5.2% of the development sets). For example, we could not automatically convert the following derivation into tags: "*locate and analyze estimated grant date fair value per ordinary share in row 7.*" The annotators wrote the instructions for these derivations in their own words, which led to inconsistency, rendering the conversion impractical. We omitted these samples and ensured that the rest could produce correct answers.

Once the dataset was ready, we randomly selected 1%, 2.5%, 5%, 10%, 25%, and 50% of the training set for the low-resource evaluation. These small samples and the development set remain the same throughout the experiment for a fair comparison. Since we do not have direct access to the test set, we only conducted a detailed evaluation with all the metrics on the development set.

4.2 Experiment setup

In addition to our reimplementation of TagOp, we created three models for evaluation. The first model (GEE: Graph Evidence Extraction) includes the GNN module for tabular graph input; the second model (NOC: Number Order Classifier) has the new number order module, but no GNN module; and the third model (GANO: Graph And Number Order) combines both methods. We chose three underlying LMs with different sizes, including RoBERTa-large (376M parameters; (Liu et al., 2019)), RoBERTa-base (136M parameters), and DistilBERT (78M parameters; (Sanh et al., 2019)). All dataset sizes, models, and LM choices make 252 training instances.

We trained the models for 50 epochs using differ-

ent learning rates for each data size, ranging from 5e-5 to 5e-3, with a batch size of 16. We used the same hyperparameter settings for each LM-data-size pair to ensure a fair comparison of all models (TagOp, GEE, NOC, and GANO). The number order classifier module in NOC and GANO is a two-layer feed-forward network with a 0.1 dropout rate. The GNN module in GEE and GANO is a single GraphSAGE layer with the same dropout rate. We used PyTorch Geometric's implementation of GraphSAGE[1] with the mean operator for the aggregator function.

This paper reports F1 scores for tabular evidence extraction and overall performance for this experiment, plus the accuracy score for number order classification. First, we will begin by comparing the performance of each proposed method to the baseline model individually, then conclude with the complete model with both modules. Due to access restrictions on the test set, the results are from the development set. However, we also included our final model's scores on the test set for comparison with the baselines. We published our source code for data preparation and all experimental settings, along with the full results involving all metrics, on our GitHub repository[2].

4.3 Tabular graph and GNN

The first experiment measures the differences the GNN module makes to the baseline model when training using different data sizes. Figure 3 compares three-run average scores between the TagOp model and our variation with the GNN module (GEE). Here we report the tabular evidence extraction scores since the component only changes this part of the model. Focusing on these scores isolates the module's effects on the outcome specific to tables (without the texts and reasoning involved), which could have implications for tabular QA.

The results on the development set show consistent advantages of the GNN module over the baseline model. Although, as anticipated, the margins are relatively lower in high-resource settings; larger models can learn and generalize tabular structures better, and more data makes recognizing patterns easier. The margins range from 0.41 to 5.05 for RoBERTa-large, 0.38 to 10.89 for RoBERTa-base, and 2.67 to 13.28 for DistilBERT.

Although the gap does not always increase with smaller data sizes, the model performs better at tabular evidence extraction when given fewer training samples. Nevertheless, the advantage is consistent across all data sizes and particularly noticeable when combined with small-scale models.

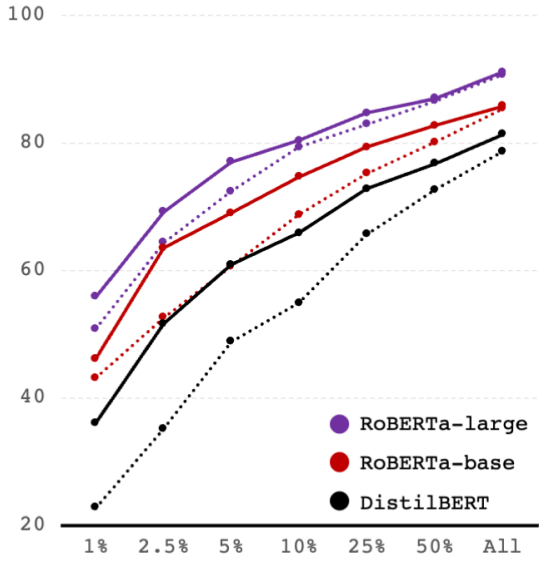

Figure 3: Three-run averages of F1 scores for tabular evidence extraction comparing the baseline model (TagOp), represented by dotted lines, to the tabular graph model (GEE), represented by solid lines. We trained the models with 1% to 100% of the training data (horizontal axis).

4.4 Number order classifier

The second experiment evaluates the number order classifier alone without the GNN component. Unlike the GNN module, the number order classifier is part of the reasoning. Therefore, we measured the accuracy of the classifier separately for NOC and GANO, then compared NOC to TagOp's overall scores. The first evaluation aims to determine how well the classifier learns and generalizes; the other measures how well the model performs with the proposed number order classifier.

When comparing NOC with TagOp (Table 1), the benefit of using our number order classifier is consistent, except for one case where we used DistilBERT with 1% of the data. In this case, because the overall F1 score is much lower than those of other settings, it tends to be less stable and reliable. Regardless, the average margins are 6.55, 4.92, and 4.27 for RoBERTa-large, RoBERTa-base, and DistilBERT, respectively.

[1]https://github.com/pyg-team/pytorch_geometric
[2]https://github.com/ichise-laboratory/finqa-gano

	Sample Size	RoBERTa-large			RoBERTa-base			DistilBERT		
		TagOp	NOC	GANO	TagOp	NOC	GANO	TagOp	NOC	GANO
1%	132	17.18	17.87	**18.25**	11.19	**14.77**	14.62	6.94	6.50	**10.30**
2.5%	330	26.26	28.19	**28.97**	26.01	30.18	**31.80**	13.19	14.32	**20.90**
5%	660	32.96	49.56	**51.33**	34.95	38.46	**41.75**	22.11	23.01	**28.64**
10%	1,321	46.92	53.87	**54.80**	41.60	45.52	**45.95**	29.10	31.48	**36.53**
25%	3,303	56.45	65.22	**66.02**	49.22	**56.67**	54.35	38.32	44.15	**46.98**
50%	6,607	65.08	70.54	**70.89**	50.51	57.62	**60.48**	45.85	50.62	**53.37**
All	12,769	72.98	**78.43**	77.78	66.52	**71.24**	70.95	58.19	63.72	**64.21**

Table 1: Three-run averages of overall F1 scores comparing the baseline model (TagOp), the model with the proposed number order classification module (NOC), and the complete model (GANO). The highest scores for each training sample size and LM are in bold.

4.5 The complete model

In the previous sections, we evaluated our proposed methods individually, and the results showed significant and consistent improvements in both modules. This third experiment measures the overall performance with both components integrated into the model. We compared our implementation of TagOp to NOC and GANO in Table 1.

GANO, which includes both modules, performs better than TagOp in every setting, regardless of the data and model size. The margins range from 1.07 to 18.37 (average 7.17) for RoBERTa-large, 3.43 to 9.97 (average 5.7) for RoBERTa-base, and 3.36 to 17.24 (average 8.15) for DistilBERT. We observed no clear difference in the margins between small and large data sizes, indicating contributions from both components; the GNN module tends to perform well with fewer training samples, while the number order classifier does the opposite.

When comparing NOC and GANO, we observed a somewhat mixed result. While GANO performs better in most settings, four cases go the opposite. The first case, RoBERTa-base with 1% of the data, sees NOC achieves a slightly higher score (0.15), which we deemed insignificant and did not pursue further investigation. The second and third cases are RoBERTa-large and RoBERTa-base with all data. These two cases indicate that the models are already capable of recognizing tabular structures given enough training data. The last case, RoBERTa-base with 25% of the data, is an outlier caused by a jump in the scale classifier's accuracy in one of the NOC training instances.

In addition to the overall scores, we also measured how well the number order classifier performed and the GNN module's effect on the classifier. Figure 4 shows that the classifier performs reasonably well on the development set (78.81% to 91.33% accurate across three LMs and model variations). The GNN module slightly harms the classifier's performance in smaller models, but overall, the accuracies are still high in both settings. It is clear from the result that the classifier is less accurate with smaller data sizes, which we anticipated since there are fewer samples to train.

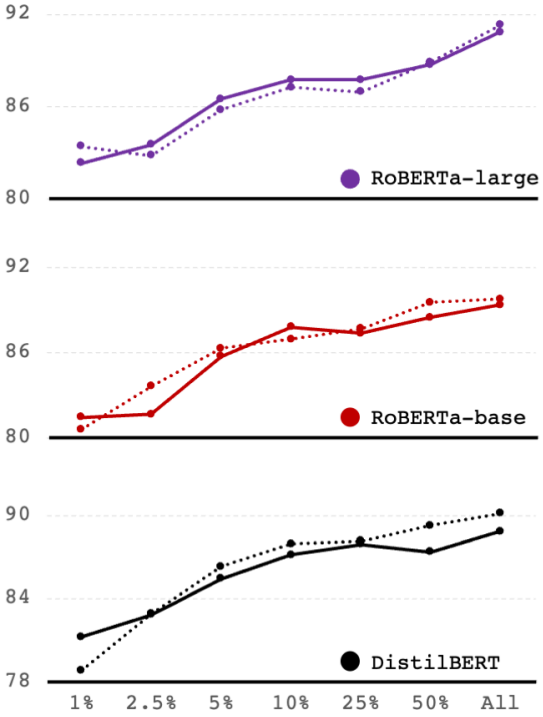

Figure 4: Three-run averages of the number order classifier's accuracy when trained with 1% to 100% of the data. Each chart compares the model with the classifier (NOC), represented by dotted lines, and the complete model (GANO), represented by solid lines.

Interestingly, while GANO almost consistently

underperforms NOC in predicting number order, as shown in Figure 4, the overall result in Table 1 indicates the opposite. We attribute this discrepancy to the magnitude of the differences the NOC module makes compared to the tabular graph module in GANO. There are 1,279 questions in the development set with answers in the tables, while only 457 questions require NOC. Our follow-up analysis shows that GANO achieved better F1 scores on tabular evidence extraction than NOC, similar to GEE and TagOp in Figure 3. The difference margins are 0.9 - 3.9 for RoBERTa-large, 0.4 - 9.6 for RoBERTa-base, and 2.4 - 15.8 for DistilBERT. GANO's large gain in the case of DistilBERT, combined with the number of questions involved, evidently outweighs its loss of up to 1.9 in number order classification accuracy.

4.6 Comparison with baselines

Although we could not conduct detailed experiments on the test set due to access restrictions, we submitted six outputs from our reimplementation of TagOp and the complete model (GANO) for evaluation. Since we implemented our data preparation algorithm differently from TagOp, we needed to evaluate our TagOp's outputs for a fair comparison. The algorithm converts the answers and derivations into evidence tags, operators, number orders, and scales to train and evaluate the models. We manually checked and corrected any questions that the algorithm could not produce the correct answers from the derivations — all of these steps we took raised TagOp's scores and, thus, need separate reporting.

Table 2 shows that GANO achieved the best scores compared to all baseline models on the development and test sets. The textual, tabular, and hybrid QA models are TagOp's baselines. According to TagOp's analysis, the authors attributed NumNet+ V2's superior performance over BERT-RC to the possibly more robust numerical reasoning capability. TaPas only learned to handle tabular data, not the hybrid table and text, and HyBrider cannot perform numerical reasoning well. Although KIQA and FinMath can outperform TagOp, GANO surpasses them significantly.

5 Discussion

5.1 Implications

The GEE's superior tabular evidence extraction scores justify its potential application in tabular QA. Our tabular graph approach is highly flexible to varying graph structures and complexities, especially when structural information is available. Since tables are typically simple and similar to each other when obtained from a collection of documents, a simple heuristic algorithm should suffice to produce such information. Although, human involvement may be necessary in supposedly rare cases where tables are highly complicated. However, since our method targets low-resource scenarios, the entire process should still be efficient.

The new number order classifier has changed our understanding of how much a hybrid QA model could achieve. We showed that the model could effectively learn to perform order-sensitive arithmetic operations with the right training strategy. The key difference here from TagOp is that the training samples need to be relevant and with minimum noise. Although the model in its current form cannot solve arbitrary math problems in natural language, as that has never been the intention, it has sufficient abilities to reason within the scope of TAT-QA, where financial documents are the objective.

5.2 Lessons learned

This section compiles our observations during implementation and experimentation as technical recommendations that we believe could be useful for future research and application. Our first recommendation concerns the use of the GNN module. As our experimental result indicated, adding the module may not always lead to the desired improvement when trained with large-scale data and LM.

The second suggestion is about the length of training. While the model could quickly recognize and generalize most tables, it took many more iterations to learn the rest. This phenomenon is not new to deep neural network models, especially LMs (Tänzer et al., 2022), and is why we chose to train for 50 epochs following TagOp's configuration. However, training for longer, e.g., 100 epochs, did not result in noticeable improvement.

Lastly, not only can multi-task learning benefit or harm the overall performance, but how information flows in the model pipeline can also significantly affect the outcome. As we experimented with different model variants before concluding the final architecture, we found that using the output representations from the GNN module for the operator and scale classifiers worsened the accuracy of both components. Thus, only the tabular evidence

	Dev		Test	
	EM	**F1**	**EM**	**F1**
Human	-	-	84.1	90.8
Textual QA				
BERT-RC	9.5	17.9	9.1	18.7
NumNet+ V2	38.1	48.3	37.0	46.9
Tabular QA				
TaPas for WTQ	18.9	26.5	16.6	22.8
Hybrid QA				
HyBrider	6.6	8.3	6.3	7.5
TagOp				
Original	55.2	62.7	50.1	58.0
Ours†				
DistilBERT	45.9	58.2	40.5	52.7
RoBERTa-base	55.5	66.6	50.0	60.3
RoBERTa-large	63.1	73.0	56.6	66.5
Hybrid & Num				
KIQA	-	-	58.2	67.4
FinMath	60.5	66.3	58.6	64.1
GANO†				
DistilBERT	51.8	64.2	46.0	58.5
RoBERTa-base	59.8	71.0	53.6	64.6
RoBERTa-large	**68.4**	**77.8**	**62.1**	**71.6**

Table 2: Comparison with the baselines on the test set. †The scores of our implementation of TagOp and our complete model (GANO) are three-run averages. TagOp's original scores are as reported in their paper.

tagger takes the GNN's output as input.

6 Conclusion

We proposed two approaches to help with a hybrid table-text QA model with numerical reasoning abilities. We added the two components to improve the baseline model's performance in low-resource settings and enhance the reasoning. The first module automatically constructs a tabular graph and uses a GNN to integrate the structure of a table into the model's pipeline. This method is beneficial to the scenario where there are limited training samples or computational resources. The second module solves the number ordering problem in certain arithmetic operations, which account for a large part of the reasoning. This module works regardless of training data or model sizes.

We conducted experiments that evaluated the proposed modules individually and collectively with four model variations, three LMs, and seven training sample sizes. Both modules demonstrated their advantages over the baseline model. The GNN module performs better with limited data and model sizes; the NOC module generally enhances the model regardless of the conditions. The experimental results also show that our tabular graph solution works with table and hybrid QA, highlighting its flexibility for future uses, potentially including other NLP tasks. The NOC module enhances the model's ability to reason on numbers, which is crucial for financial QA and other application domains.

7 Acknowledgment

This work was partially supported by the New Energy and Industrial Technology Development Organization (NEDO).

References

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021. Muppet: Massive multi-task representations with pre-finetuning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 5799–5811.

Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q Tran, Dara Bahri, Jianmo Ni, et al. 2022. ExT5: Towards extreme multi-task scaling for transfer learning. In *Proceedings of the International Conference on Learning Representations*.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *Proceedings of the International Conference on Learning Representations*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. 2020. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics, the Conference on Empirical Methods in Natural Language Processing*, pages 1026–1036. Association for Computational Linguistics.

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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, 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 *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 2368–2378.

Chris Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. 2021. Efficiently identifying task groupings for multi-task learning. In *Proceedings of the 35th Conference on Neural Information Processing Systems*, pages 27503–27516.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of Advances in Neural Information Processing Systems, the 31st Conference on Neural Information Processing Systems*, volume 30, pages 1024–1034.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (GELUs). *arXiv:1606.08415*.

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.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1821–1831.

Chenying Li, Wenbo Ye, and Yilun Zhao. 2022. FinMath: Injecting a tree-structured solver for question answering over financial reports. In *Proceedings of the 13th Conference on Language Resources and Evaluation*, pages 6147–6152.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Rungsiman Nararatwong, Natthawut Kertkeidkachorn, and Ryutaro Ichise. 2022. KIQA: Knowledge-infused question answering model for financial table-text data. In *Proceedings of the 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 53–61.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd Workshop on Very Large Corpora*.

Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. NumNet: Machine reading comprehension with numerical reasoning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2474–2484.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Michael Tänzer, Sebastian Ruder, and Marek Rei. 2022. Memorisation versus generalisation in pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 7564–7578.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 6442–6454.

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, pages 3277–3287.