Applying Question Decomposition to Numerical Reasoning over Multi Hierarchical Data

Sashank Kakaraparty
George Mason University
skakarap@gmu.edu

Madhukar Reddy Vongala
George Mason University
mvongala@gmu.edu

Sai Karthik Dindi
George Mason University
sdindi@gmu.edu

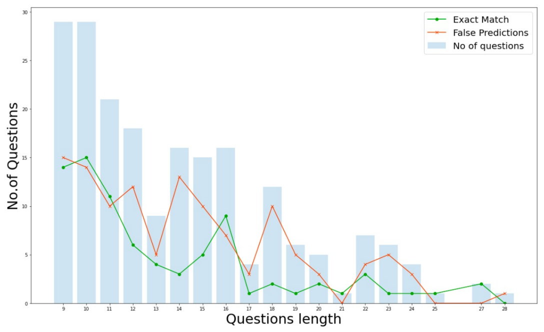
1 Introduction

Question Answering is one of the most popular forms of NLP we interact with, in day-to-day lives. Question Answering is used in Chatbots answering questions on customer support roles, it's used everytime we search for something on a search engine and it gives you an answer directly without having to visit any website. There has also been the use of QA techniques to answer questions with information embedded in tables, which has several use cases in areas where data is stored in tables (example: Financial reports, Product Specifications, etc.) where retrieval is required. Taking it a step further, there are use cases where we not only retrieve information but also perform number reasoning over such tabular and text data together, which is the area we have studied our problem in and have worked towards solving.

In particular, we have focused on improving the baseline on Numerical Reasoning over Text and Tabular data (Zhao et al., 2022) which includes tables that are multi-hierarchical which adds more complexity to the QA process. The promise of improvement in this area is encouraging as it will have a large impact on how large amounts of tabular information can be consumed, which is the case in several areas such as finance and industrial engineering, where large hierarchical tables containing lots of information are the norm.

Our proposed approach in this attempt to improve upon this problem comes from a detailed error-analysis of the current baseline, where we recognize that performance deteriorates as the question gets longer, or in other words has "multiple-hops", which would mean it has to retrieve information about more that one entity from more than one area, to be able to answer the question. This analysis shows that the baseline approach is relatively stronger at answering questions that have "single-hops", and are usually shorter.

Hence, our proposed approach is a result of this analysis on the existing methods and if described in the simplest words, tries to convert these hard questions into several simpler questions. In more detail, we try to convert a single question that requires multiple-hops to get the required information into multiple questions that require single-hops and can utilise each of those results to build up to the initial question. Formally speaking, we attempt to decompose a question into simpler questions which we have observed is where our model is relatively stronger at giving us the right result. We then recompose the results from each of these new sub questions and arrive at answering the original question.



Although we will demonstrate in further detail, our work and results in the coming sections, question decomposition is definitely a promising approach in our area of study. It furthermore makes sense, because it's common in numerical reasoning questions to have more than one entity, requiring us to perform information retrieval over multiple hops. A more detailed study of the results in the later sections will also reveal that our approach to how we decompose questions might have some inherent assumptions and would require more finetuning over data specific to our domain. To summarise, the approach of decomposing questions to improve performance is promising, when coupled with a well built and domain aware decomposition pipeline.

Code we've developed as part of this project is publicly available at https://github.com/iamsashank09/question-decomposition-multihiertt

2 Related Work

Complex Tabular and Hybrid Question Answering; There's been considerable work in this area, which includes classic datasets such as SQuAD (Rajpurkar et al., 2016) and the newer SQuAD 2.0 (Rajpurkar et al., 2018) and others which contain simple structured tables and knowledge bases. However, the system becomes complex as we move to complex hierarchical tables in works such as Air-QA (Katsis et al., 2021) and HiTab (Cheng et al., 2022), while also foraying into hybrid data, where there are both tables and text data which are required to answer the questions such as FinQA (Chen et al., 2021), TAT-QA (Zhu et al., 2021) and HybridQA (Chen et al., 2020) and MultiHiertt (Zhao et al., 2022), which brings multi-hierarchical table data alongside text data, making it a hybrid multi-hierarchical QA problem. Numerical Reasoning over this data has explored several techniques, but the best results have been seen by models proposed in MultiHiertt (Zhao et al., 2022) and FinQA (Chen et al., 2021) over hybrid data (table + text).

Question Decomposition; In multi-hop datasets, the evidence for answering the question is scattered across multiple tables/paragraphs. Some datasets like HotpotQA, although built for Reading Comprehensions, are explicitly built to test performance of QA models on multi-hop questions. There have been several methods in decomposing questions, everything from defining rule based approaches to decompose questions, such as (Fan et al., 2012) which although simpler and more reliant on semantic parsing, becomes harder to adapt outside the modelled domain. In (Iyyer et al., 2016) the authors crowd-source decompositions and train a network to predict the answer based on the simple questions. In terms of more supervised approaches, (Min et al., 2019) proposes an approach similar to (Talmor and Berant, 2018) which has shown great results on datasets such as HotpotQA in the domain of QA over Reading comprehensions. More recently, (Perez et al., 2020) proposed an unsupervised approach where the authors teach a discriminator to learn the difference between simple and complex questions and define a cost function to maximise similarity between the parent question and the subquestions, while at the same time decrease similarity among the subquestions.

In terms of related work, our work is most related to (Zhao et al., 2022) and (Min et al., 2019) in the sense that we have adapter our modelling pipeline based on the MT2NET Modelling pipeline proposed by (Zhao et al., 2022), and our question decomposition technique is inspired from the "Bridging" type decomposition proposed by (Min et al., 2019). These two works have a direct influence on our approach, as we try to combine their proposed solutions to enhance the performance on Hybrid Multi-Hierarchical Question Answering. .

3 Approach

Let's start with the basics of both our key features - Numerical Reasoning + Question Answering and Question Decomposition, after which we go into the details of how each of these fits into our workflow.

In Question Answering, a system answers questions based on a corpus, which in our case is a combination of Text and Tables (hence, Hybrid) and an additional feature of our tables being that they are hierarchical.

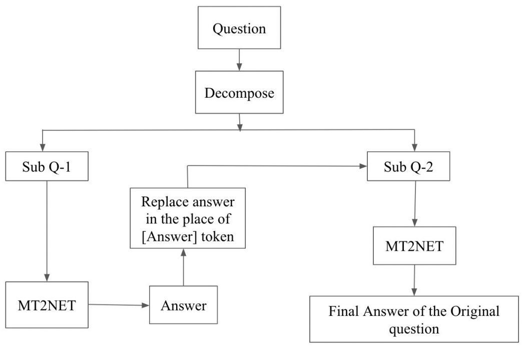
In Question Decomposition, a system breaks down a question into several smaller questions, based on a specific criteria, which in our case is the complexity of our question i.e., we break a complex question into smaller questions.

In our approach, we propose a pipeline for Multi-hierarchical Numerical Reasoning using Question Decomposition, a three step process:

- First, our modelling pipeline decomposes the original question into two sub-questions where the second question is based on the first question.

- We pass the first sub-question through our QA models and generate the output, which is the "SQ-1 ANS" indicating that it is the answer to the first subquestion.

- Following on, we fill this "SQ-1 ANS" into our second subquestion and pass it to our QA models again, which generates the answer, which is also the answer to our original question.


Question
Decompose
Sub Q-1
Sub Q-2
Replace answer in the place of [Answer] token
MT2NET
MT2NET
Answer
Final Answer of the Original question

3.1 Numerical Reasoning over Multi Hierarchical Tables A.K.A QA Module

Our QA module in itself contains 4 submodules, which each serve a different purpose in the grand scheme of responding to numerical reasoning questions, let's break each one of them down for starters -

3.1.1 Retriever Module:

The first step of the QA process is to select the top-n facts which are relevant to the question being asked. This is a key step, because the rest of the pipeline will depend on these "n" facts/information to find the answer to the question. When picking the top "n" facts, it processes all the facts available for the question and gives each one of them a score, which are then ordered to prioritise the facts with the highest scores.

In terms of implementational details, the retriever module is a BERT based model with two linear layers at the end. It is finetuned on a large corpus to identify relevant facts from a large collection of facts, based on what the question is asking. The model goes through each and every sentence in our dataset and tags them either as a "1" or "0". The variant of BERT we use is RoBERTa's base version.

3.1.2 Question Classification Module:

While the retriever is running parallely, we also try to classify the question between two types - "SPAN" and "PROGRAM GENERATION". A "SPAN" question is one which possibly has an answer in the inline text evidence, vs the "PROGRAM GENERATION" would require us to perform one or more arithmetic operations to arrive at the answer. This classification is key, because this decides the next step in the pipeline.

In terms of implementation details, a BERT based model with a Sequence Classification architecture is finetuned on an annotated corpus of questions classified as one of two categories. We used the RoBERTa implementation with the output layers being a binary classification function.

3.1.3 Span Selection module:

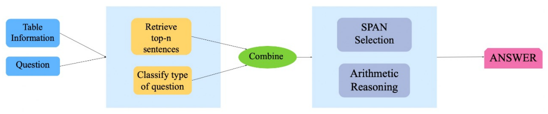
If a question is classified as "span_selection", then it is passed to the Span Selection Module, but before which we use a language model to encode the retrieved sentences. The data that the span selection model works with, is questions that are expected to have answers in the text and table descriptions, without having to perform further arithmetic calculations. The module aims to select the predicted span candidate, which is a span of retrieved sentences. The answer probability is defined as the product of the probabilities of the start and end positions in the retrieved evidence.

In terms of implementation details, This model is based on the T5 Model - Text-toText Transfer Transformer model, an encoder-decoder model, which we learnt has advantages in certain areas over BERT variants. The base encoder-decoder T5 is followed by a linear layer to the end which outputs logits in the range of the tokens. which we are choosing among. In this approach, when we infer, the input text is tokenized and then the token which the model believes is the answer carries the highest score.

3.1.4 Program Generation Module:

If a question is classified as "arithmetic", then it is passed to the Program Generation Module, but before which we use a language model to encode the retrieved sentences. The sub-module aims to generate the executable program to answer the question using an LSTM for decoding. At each decoding step T, the LSTM can generate one token between Number (Ex: A number from the retrieved data), A predefined operator (Ex: An operator like '+, -, * ..), or tokens already generated in the previous steps. After it is generated, the sub-module will execute the generated programs and get the predicted answer finally.

In the end, we combine the results from both the modules, if there are more than one question being sent through the pipeline.



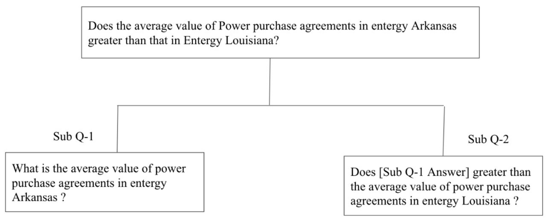
3.2 Question Decomposition

The goal of model decomposition is to convert a multi-hop question into multiple single hop questions, which is challenging because annotations are hard to come-by. Whereas, another option is to generate the decompositions word-by-word, which is a tricky task as well. In the approach we use, we work around with a completely different intuition. The key intuition on which the entire approach is based, is that each sub-question can be formed by copying and slightly editing a key span from the original question! In essence, we try to conduct a span prediction over the original question.

In practice, when a new question is passed, it identifies the spans in the question. If two entities are extracted as spans, then the question can be converted to two sub-questions. In terms of implementation details, we have used BERT as the Language Model for span selection.

Now, onto how we use the decomposed questions. As described earlier, we generate the decompositions and pass the first subquestion to our QA pipeline, from which we use the result to fill the second subquestion.

Earlier in our implementation, we were concerned about the recomposition aspect, but once we understood the author's bridging model, we were able to remove the need for recomposition and instead built questions that would use the previous information.


Does the average value of Power purchase agreements in entergy Arkansas greater than that in Entergy Louisiana?
Sub Q-1
What is the average value of power purchase agreements in entergy Arkansas ?
Sub Q-2
Does [Sub Q-1 Answer] greater than the average value of power purchase agreements in entergy Louisiana ?

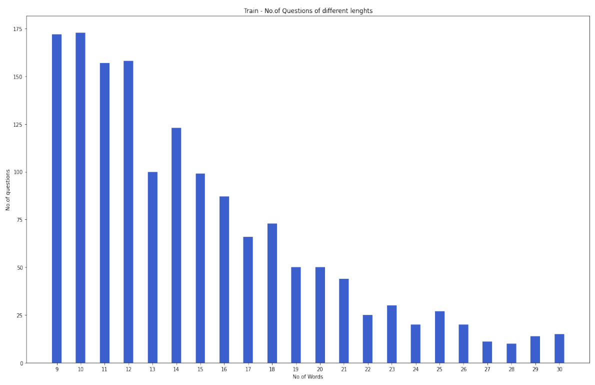
4 Experiments

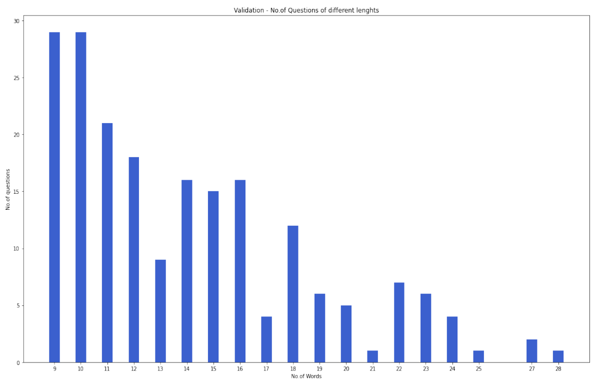
4.1 Datasets

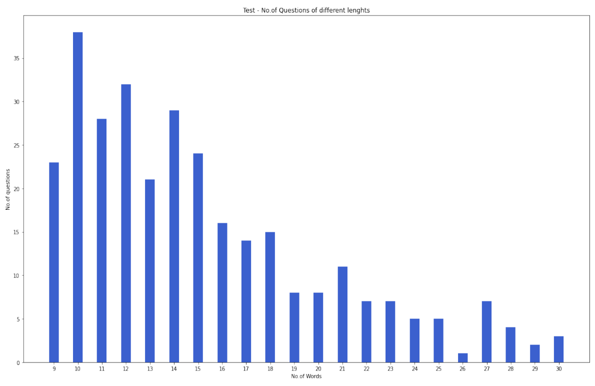
The dataset have have used in our experiments is the MultiHiertt Dataset from the work of (Zhao et al., 2022) where they constructed a large-scale benchmark. This dataset has documents that contain multiple tables and long unstructured texts, where most tables are hierarchical. The reasoning process required for each question is more complex and annotations of the reasoning process are also provided. Overall, it is the first dataset to study numerical reasoning questions over hybrid data containing multiple hierarchical tables. It contains questions which are on the lines of WHAT, HOW, WHICH, WHEN, IF (in that order of their frequency from high to low). In addition, this is the dataset our baseline work (detailed below) has used, making it a convenient comparison.

4.1.1 Data Analysis

We dig deeper into the dataset and found out that the major part of the questions in the train set have questions consists of words under 13 whereas, the validation and test set has more number of questions with more than 13 words.







From the above figures, it is clear that the model is strong in the area where the number of words in a question are less than or equal to 13. This again drives us in the direction of decomposing the long questions into multiple short questions since the model can answer the short questions well.

4.2 Baselines

The baseline work we are expecting to replicate is the MultiHiertt: Numerical Reasoning over Multi Hierarchical Tabular and Textual Data (Zhao et al., 2022), they benchmark the model on data with plain text, plain tables, combination of both in hierarchical settings with multiple step questions and other combinations, where the results obtained are the following.

Performance Breakdown	**EM**	**F_1**
Regarding supporting facts coverage		
text-only questions	49.26	53.29
table-only questions	36.77	38.55
w/ ≥ 2 tables	24.32	24.96
table-text questions	33.04	35.15
w/ ≥ 2 tables	21.04	23.36
Regarding numerical reasoning steps		
1 step	43.62	47.80
2 steps	34.67	37.91
3 steps	22.43	24.57
>3 steps	15.14	17.19
Full Results	**36.22**	**38.43**

4.3 Experimentation Setup

From Fig-1, it is clear that the number of false predictions increases if the number of words in the questions is greater than 13. So, we divided the questions into two sets to decompose:
1) Short questions: Questions having number of words less than or equal to 13.
2) Long questions: Questions having number of words greater than 13. We decompose the long questions.

NOTE: Here, we recognize that we are making an assumption that longer questions are more complex, vs shorter questions are simpler. This is a general assumption made through observation.

4.4 Evaluation Metrics

We take in the predicted and the gold answers to normalise them i.e preprocess the answers like remove punctuation (if not a number), lowering the text, etc.,. The predictions and gold answers are then converted to what are called the prediction bags and gold bags. These bags contain the predictions as a tuple of predictions and tokens set of that prediction.

To compute the exact match, the predictions in the bags and length of those predictions are compared. For the F1 score, gold and predicted answer sets are taken and the optimal 1-1 alignment between is found to compute the precision and recall based on the number of intersecting tokens in the sets, which are in turn are used to compute the F1 score.

$$Precision = \frac{len(intersection(prediction\_bag, gold\_bag)}{len(predicted\_bag)}$$

$$Recall = \frac{len(intersection(prediction\_bag, gold\_bag))}{len(gold\_bag)}$$

F1 value for each pair of (prediction, gold answer) is computed and then their mean is computed to get the overall F1 score.

4.5 Results and Discussion

We have a attached a few outputs of our modelling pipeline's performance below, for observation.

Example 1: Where we got it slightly better than the baseline.

Question-1: In the year with the largest amount of Office Net Charge-offs in table 3, what's the increasing rate of Office Net Charge-offs in table 3?

Sub Q1: Which year with largest amount of office net charge-offs in table 3?

Sub Q2: in [Answer] in table 3, what is the increasing rate of office net charge-offs in table 3?

Original Answer: 0.53846
Answer by MT2NET: -0.53846
Answer by Our Model: 0.53846

Example 2: Where we got it way better than the baseline.

Question-2: what is the sum of the cost of sales in the years where net sales is greater than 1000? (in million)

Sub Q1: which years where net sales is greater than 1000? (in million)

Sub Q2: what is the sum of the

cost of sales in [ANSWER]

Original Answer: 1867.0
Answer by MT2NET: 1599.0
Answer by Our Model: 1867.0

Example 3: Still wrong, but closer to the answer.

Question-3: Does the average value of Power purchase agreements in entergy Arkansas greater than that in Entergy Louisiana?

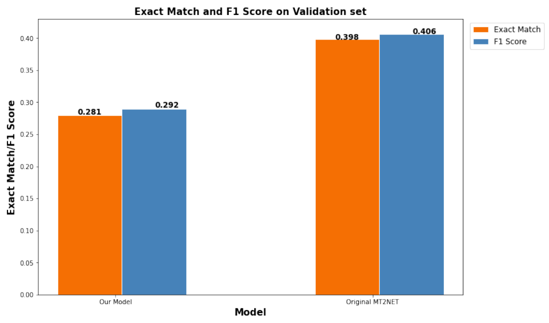
Sub Q1: What is the average value of power purchase agreements in entergy Arkansas ?

Sub Q2: does [Answer] greater than the average value of power purchase agreements in entergy Louisiana ?

Original Answer: Yes
Answer by MT2NET: -1
Answer by Our Model: No

As we get to the numbers, things start to get interesting..

We ran all the development/validation data through our pipeline and this is how it performed -

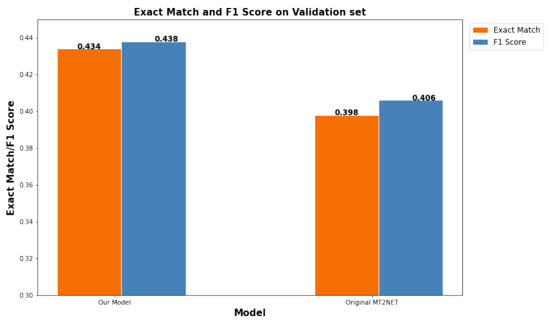


As you can see, the metrics of our model fall drastically, almost cut in half.

After taking a closer look, we understood that this is the result of decomposition hurting perfectly well performing questions, which was counter productive. (This also goes back to our assumption about all long questions being complex questions)

To understand our model's performance, We took only the questions in the validation set which were incorrectly predicted by the MT2NET without decomposition and passed them to our pipeline. After predicting the answers to those questions, we added them to the questions correctly predicted by MT2NET and our model without decomposition.

Here are the results we saw -



As we can see, the model does predict correctly on questions that were earlier not being predicted correctly, showing that decomposition does have the potential to improve the performance of the QA Pipeline.

We observed a similar dip in performance when we uploaded our results to Codalab to evaluate our performance in the test set.

Exact Match : 28.76
F1-Score : 28.93

Whereas the author's results stand at:

Exact Match : 38.63 **F1-Score** : 39.02

5 Conclusion

In conclusion, we have observed that question decomposition will definitely help improve the performance of Numerical Reasoning over Hybrid data, however, the big catch seems to be that the implementation has a long way to go as the numbers show. Some of the items that have held us back, are factors such as the decomposition models not actually being trained towards financial data, which meant it is a zero-shot approach in those aspects. We need to have a more robust decomposition pipeline, in order to fulfill the promise that question decomposition offers, and some of which we have provided in our future works below.

6 Future Work

Here are a few idea we would implement to better the experiments:

- It is not a good idea to decompose every long question, as evident by the results, some questions may not need decomposition. We would like to build a discriminator model that can decide whether a question requires decomposition, evaluating if the question is complex or simple, something similar to what the authors of Perez et al. (2020) have explored.

- One of the reasons our decomposition model does not perform extremely well on our MultiHiertt data, is because it was trained on the HotPot QA (**?**), which does not specialize in finance, so we would like to finetune this current decomposition model with our own hand-annotated data, which would help decompose better.

- During our error-analysis, we recognized that a lot of questions in this dataset, understandably are comparison questions i.e., use terms such as greater than, smaller than, higher, lower etc., which are currently not being handled by MT2NET's program generation module. Hence, adding the comparison capability at some stage would help improve performance.

References

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036, Online. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110, Dublin, Ireland. Association for Computational Linguistics.

J. Fan, A. Kalyanpur, D. C. Gondek, and D. A. Ferrucci. 2012. Automatic knowledge extraction from documents. *IBM Journal of Research and Development*, 56(3.4):5:1–5:10.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2016. Answering complicated question intents expressed in decomposed question sequences.

Yannis Katsis, Saneem A. Chemmengath, Vishwajeet Kumar, Samarth Bharadwaj, Mustafa Canim, Michael R. Glass, Alfio Gliozzo, Feifei Pan, Jaydeep Sen, Karthik Sankaranarayanan, and Soumen Chakrabarti. 2021. AIT-QA: question answering dataset over complex tables in the airline industry. *CoRR*, abs/2106.12944.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600, Dublin, Ireland. Association for Computational Linguistics.

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. *CoRR*, abs/2105.07624.