AIT-QA: Question Answering Dataset over Complex Tables in the Airline Industry

Yannis Katsis[1], **Saneem Chemmengath**[1], **Vishwajeet Kumar**[1], **Samarth Bharadwaj**[2*], **Mustafa Canim**[1], **Michael Glass**[1], **Alfio Gliozzo**[1], **Feifei Pan**[3], **Jaydeep Sen**[1], **Karthik Sankaranarayanan**[1], **Soumen Chakrabarti**[4]

[1] IBM Research [2] Microsoft [3] Rensselaer Polytechnic Institute [4] IIT Bombay

yannis.katsis@ibm.com, saneem.cg@in.ibm.com, vishk024@in.ibm.com, sabharadwaj@microsoft.com, mustafa@us.ibm.com, mrglass@us.ibm.com, gliozzo@us.ibm.com, panf2@rpi.edu, jaydesen@in.ibm.com, kartsank@in.ibm.com, soumen.chakrabarti@gmail.com

Abstract

Table Question Answering (Table QA) systems have been shown to be highly accurate when trained and tested on open-domain datasets built on top of Wikipedia tables. However, it is not clear whether their performance remains the same when applied to domain-specific scientific and business documents, encountered in industrial settings, which exhibit some unique characteristics: (a) they contain tables with a much more complex layout than Wikipedia tables (including hierarchical row and column headers), (b) they contain domain-specific terms, and (c) they are typically not accompanied by domain-specific labeled data that can be used to train Table QA models.

To understand the performance of Table QA approaches in this setting, we introduce AIT-QA; a domain-specific Table QA test dataset. While focusing on the airline industry, AIT-QA reflects the challenges that domain-specific documents pose to Table QA, outlined above. In this work, we describe the creation of the dataset and report zero-shot experimental results of three SOTA Table QA methods. The results clearly expose the limitations of current methods with a best accuracy of just 51.8%. We also present pragmatic table pre-processing steps to pivot and project complex tables into a layout suitable for the SOTA Table QA models. Finally, we provide data-driven insights on how different aspects of this setting (including hierarchical headers, domain-specific terminology, and paraphrasing) affect Table QA methods, in order to help the community develop improved methods for domain-specific Table QA.

1 Introduction

The tabular data format is commonly used in digital documents such as PDFs and HTMLs to store semi-structured information (Canim et al., 2019; Zhang and Balog, 2018; Pasupat and Liang, 2015). Due to the rich content found in tables, many works have looked into extracting information out of tables (Burdick et al., 2020) and leveraging it for various NLP tasks, such as answering questions over tables (Cafarella et al., 2009; Sun et al., 2019; Shraga et al., 2020a,b). The quality of answers depends on first, high quality extraction of tables out of documents (aka *Table Extraction*); second, retrieval of relevant tables for a given natural language question or keyword query (aka *Table Retrieval*); and finally, identification of the relevant cells over the retrieved tables (aka *Table QA*). Most recently, transformer-based pre-trained architectures such as TaBert (Yin et al., 2020), TaPas (Herzig et al., 2020), and RCI (Glass et al., 2021) have been proposed to tackle the Table QA task by identifying table cells containing the answer to a given question. These models have been shown to exhibit very high accuracy in Table QA. However, the results are based on training and testing the proposed techniques on *open in-domain* datasets, built on top of Wikipedia tables, such as the WikiTableQuestions (Pasupat and Liang, 2015) and WikiSQL (Zhong et al., 2017) datasets.

Based on our experience in designing and implementing industrial table processing approaches (Burdick et al., 2020), open-domain web tables typically exhibit much simpler structures than the ***complex table structures*** found in *domain-specific* scientific or business documents. For instance, consider the sample question-table pair from our proposed airline dataset shown in Figure 1. The table contains both column headers and row headers (i.e., descriptors of columns and rows, respectively) and both of them are hierarchical in nature. Moreover, answering a question often requires reasoning on such complex column/row header hierarchies. For instance, finding the requested main-

[*] Work done while author was working at IBM.

Proceedings of NAACL-HLT 2022: Industry Track Papers, pages 305 - 314
July 10-15, 2022 ©2022 Association for Computational Linguistics

Question: What was the reported mainline RPM for American Airlines in 2017 ?

Hierarchical column headers

Hierarchical row headers

	Year Ended December 31,		
	2017	2016	2015
Mainline			
Revenue passenger miles (millions) [a]	201,351	199,014	199,467
Available seat miles (millions) [b]	243,806	241,734	239,375
Passenger load factor (percent) [c]	82.6	82.3	83.3
Yield (cents) [d]	14.52	14.02	14.56
Passenger revenue per available seat mile (cents) [e]	11.99	11.55	12.13
Operating cost per available seat mile (cents) [f]	12.96	11.94	12.03
Aircraft at end of period	948	930	946
Fuel consumption (gallons in millions)	3,579	3,596	3,611
Average aircraft fuel price including related taxes (dollars per gallon)	1.71	1.41	1.72
Full-time equivalent employees at end of period	103,100	101,500	98,900
Total Mainline and Regional			
Revenue passenger miles (millions) [a]	226,346	223,477	223,010
Available seat miles (millions) [b]	276,493	273,410	268,736
Passenger load factor (percent) [c]	81.9	81.7	83.0

Figure 1: Question-table pair in AIT-QA, showing the complex structure of tables in the dataset. The cell containing the answer is shown in blue and its hierarchical column/row headers are shown in orange and green, respectively

line Revenue Passenger Miles (RPMs) (which are contained in the blue cell) requires understanding that the cell has two hierarchical row headers "Mainline" and "Revenue passenger miles" (shown in green). Ignoring row headers or not reasoning on the row header hierarchy may lead to wrong results. For instance, if we simply searched for cells with a flat row header containing "Revenue Passenger Miles", we may mistakenly return value 226,346 appearing further down the table (which corresponds to the RPMs of *total* operations, instead of the requested *mainline* operations). In contrast, web tables in open-domain Table QA datasets, such as WikiTableQuestions or WikiSQL, exhibit significantly simpler structures. Such tables do not contain row headers and only have a single column header, closely resembling relational database tables.

Moreover, while open-domain datasets capture common entities, such as locations, person names, etc, which often appear in Wikipedia articles, they typically lack ***domain-specific vocabulary*** that one encounters in scientific or business documents (such as the “Revenue Passenger Miles" above).

Finally, from a deployment point of view, when clients bring their own domain-specific documents, they typically ***do not provide domain-specific labeled data*** to train Table QA models. This comes in contrast to existing Table QA evaluations on open-domain data, where tested models are first trained using large amounts of open-domain training data.

Based on the above, domain-specific Table QA exhibits major differences from open-domain settings: Domain-specific documents include *more complex table structures* and *specialized vocabulary*, and are *not accompanied by domain-specific labeled data*.

To understand whether existing Table QA approaches support these settings, we create AIT-QA; a domain-specific Table QA dataset, where tables are extracted from financial documents in the airline industry. The majority of the tables exhibit a complex structure, including hierarchical row and column headers, as well as airline-specific terminology. Finally, we build the dataset as a test set to reflect the lack of domain-specific labeled data and encourage works on zero/few-shot learning. To the best of our knowledge, this is the first Table QA dataset that includes and explicitly encodes such complex table layouts and domain-specific table contents. Our experiments with three state-of-the-art Table QA models show that existing models struggle to support this setting, yielding an accuracy of at most 51.8%. We hope that this dataset and associated insights will help the community better support domain-specific Table QA in the future.

This work makes the following contributions:

A complex and domain-specific Table QA dataset called *AIT-QA (Airline Industry Table QA)*, created by human annotators based on 10-K financial reports of major airline companies. The questions are created based on the content of tables appearing in the 10-K reports, as well as KPIs (Key Performance Indicators); i.e., important metrics tracked by analysts in the airline industry. The dataset has been made publicly available under a CDLA-Sharing-1.0 license at `https://github.com/IBM/AITQA`.

Experimental evaluation of state-of-the-art Table QA models on AIT-QA, demonstrating that high performance on open-domain datasets does not

Dataset	Year	Table only	Wikipedia	Hierarchical Column Headers	Hierarchical Row Headers
WikiTableQuestions (Pasupat and Liang, 2015)	2015	✓	✓	✗	✗
TabMCQ (Jauhar et al., 2016)	2016	✓	✗ (Science Exam)	✗	✗
WikiSQL (Zhong et al., 2017)	2017	✓	✓	✗	✗
FeTaQA (Nan et al., 2022)	2021	✓	✓	✗	✗
HybridQA (Chen et al., 2020)	2020	✗	✓	✗	✗
OTT-QA (Chen et al., 2021)	2021	✗	✓	✗	✗
TAT-QA (Zhu et al., 2021)	2021	✗	✗ (Finance)	✗	✗
AIT-QA (this work)	2021	✓	✗ (Airlines)	✓	✓

Table 1: Comparison of AIT-QA to other Table QA datasets

guarantee similar performance on domain-specific datasets containing complex tables, further motivating the need for a domain-specific Table QA dataset.

A novel data pre-processing technique for existing Table QA models, which improves their performance on datasets with complex table structures. This is achieved by translating complex table structures (incl. hierarchical row and column headers) to simpler structures resembling the structure of the tables on which such approaches have been trained.

2 Related Work

Prior work on leveraging tables to answer questions studied two tasks: (a) *Table retrieval*; i.e., given a corpus of tables, identify the table containing the answer to a question, and (b) *Table QA*; i.e., given a single table containing the answer, find this answer. We next discuss datasets for Table QA, which is the focus of this work.

The most commonly used Table QA datasets include WikiTableQuestions (Pasupat and Liang, 2015), WikiSQL (Zhong et al., 2017), and TabMCQ (Jauhar et al., 2016). Out of them, the first two are based on Wikipedia. The third, contains manually-curated general knowledge tables created from the Regents 4th-grade exam. While it is domain-specific, the included tables have a very peculiar structure (with table rows containing entire natural language sentences that have been split into columns), which in our experience is not representative of tables appearing in most domains. Recently, Nan et al. (2022) proposed FeTaQA; another Wikipedia-based dataset but with answers that are long free-form sentences (instead of short answers found in prior datasets).

Finally, the last couple of years saw the introduction of three multi-hop QA datasets: *HybridQA* (Chen et al., 2020), *OTT-QA* (Chen et al., 2021), and *TAT-QA* (Zhu et al., 2021). In these datasets finding an answer requires reasoning not only on tables but across both tables and associated text. Out of them, HybridQA and OTT-QA are both based on Wikipedia. On the other hand, TAT-QA, is based on data extracted from financial reports, making it the most similar to our proposed AIT-QA dataset.

However, while the TAT-QA paper mentions complex table structures (including row headers), the resulting dataset does not include explicit annotations of row and column headers (not to mention hierarchies thereof). Without explicit annotations of such headers, not only is it hard to understand the complexity of the included tables (e.g., the Appendix of (Zhu et al., 2021) points to the absence of column header hierarchies in TAT-QA), but it also makes it harder to understand the effect of table complexity on the performance of Table QA algorithms. Instead, our proposed AIT-QA treats hierarchical column and row headers as first-class citizens and is to the best of our knowledge the first *domain-specific* Table QA dataset that contains *explicit annotations of complex table structures, including hierarchical row and column headers*. Table 1 summarizes the discussed Table QA datasets.

3 Dataset

We next outline the process followed to generate AIT-QA, from data acquisition and preparation to question annotation and table header identification.

Data Acquisition. AIT-QA is based on 10-K forms; comprehensive annual reports that publicly traded companies file with the U.S. Securities and Exchange Commission (SEC). For this dataset, we focused on the airline industry and retrieved recent 10-K forms of all 5 airlines included in the Standard & Poor's 500 (S&P 500) stock market index[1]. Covered airlines include: Alaska Air Group (ALK), American

[1] https://en.wikipedia.org/wiki/List_of_S%26P_500_companies

Airlines Group (AAL), Delta Air Lines Inc. (DAL), Southwest Airlines (LUV), and United Airlines Holdings (UAL). The 10-K forms were downloaded through the publicly accessible SEC EDGAR online system[2] in HTML form.

Data Preparation and Cleaning. While the downloaded 10-K forms encode tables using standard HTML tags, the tables are formatted with human consumption in mind. As such, table rows/columns/cells are used for the table to be neatly rendered on the screen and/or paper and they do not always correspond to the table's logical structure. In particular, we found that tables often contain extraneous rows/columns (introduced to allow for more space between table elements). Moreover, the contents of a single logical cell are often split into multiple physical cells, to allow for better vertical alignment of the information within a table. For instance, cells containing a currency symbol and negative monetary amounts such as $\$(1{,}234)$, are often split into three physical cells | $ | (1,234 |) | so that the currency symbols and numbers align with other similar contents across rows. To separate these formatting decisions from the logical structure of the table, we post-processed the downloaded HTML files to remove extraneous rows/columns and merge back components of logical cells originally split into multiple cells. Processing was done through a combination of scripts and manual error correction.

Question Annotation. The cleaned 10-K forms were given to 8 co-authors of this paper to generate question-answer pairs over tables appearing on the forms. To capture questions of particular interest to domain experts in the domain, while ensuring a diversity of question topics, we asked annotators to provide two types of questions:

KPI-driven questions: These are questions that inquiry about Key Performance Indicators (KPIs), which are metrics of particular interest to analysts in the airline industry. Annotators were provided with a list of KPIs along with common synonyms to ensure that the questions capture not only the topic of interest but also use the correct vocabulary. Then they were instructed to search the document for mentions of KPIs within tables and create corresponding questions. Thirteen KPIs were used in total, each with three variants, depending on whether it referred to the airlines' mainline, regional, or total operations.

Table-driven questions: While KPI-driven questions capture common metrics tracked by analysts, they can be limiting for two reasons: First, there is a small number of KPIs and second, given their domain importance, they often appear within a small set of tables. Hence, limiting ourselves to such questions would lead to a non-diverse dataset. To avoid this issue, annotators were asked to also provide questions that inquired about other concepts appearing within the input tables. To create such questions, annotators browsed through the tables in the documents and wrote questions that could be answered by them.

After an initial set of question-answer pairs was collected, annotators were also asked to generate paraphrases. While creating paraphrased questions, annotators were given access to the set of question-answer pairs collected in earlier stages and asked to pick a subset of questions to paraphrase. This leads to the second major dimension along which questions in AIT-QA can be classified: ***Original questions*** (Questions collected during the initial annotation) and ***Paraphrased questions*** (Questions generated as paraphrases of original questions).

Finally, in all stages of the annotation process, annotators were asked to keep track of additional metadata indicating whether a question relied on the hierarchy of row headers to be answered. A question relies on the hierarchy of row headers when in order to be unambiguously answered, one has to consult not only the row header that appears on the same row as the answer, but also row headers appearing on higher levels of the hierarchy. For instance, the question in Figure 1 depends on the row header hierarchy, as ignoring the hierarchy may lead to an incorrect answer, as explained in the introduction. Based on these metadata, questions in the dataset can be differentiated across a third dimension into: ***Row header hierarchy questions*** (Questions whose answer relies on the row header hierarchy) and ***No row header hierarchy questions*** (Questions whose answer does not rely on the row header hierarchy).

For each question-answer pair, annotators provided the question, the table cell where the answer appears, as well as metadata indicating the classification of the question along the three aforementioned dimensions. For the first version of the dataset, we focus on *lookup* questions - i.e., questions where the answer appears within table cells and does not require aggregate operations (such as min/max/sum/count) to be returned (Glass et al., 2021), leaving the expansion of the dataset with aggregate questions as future work. Annotation was carried out using a custom-built Table QA annotation tool (see Appendix A for more details). Finally, the collected question-answer pairs

[2] https://www.sec.gov/edgar.shtml

Question Type	Count (%)
KPI-driven questions	145 (28%)
Table-driven questions	370 (72%)
Original questions	439 (85%)
Paraphrased questions	76 (15%)
Row header hierarchy questions	146 (28%)
No row header hierarchy questions	369 (72%)

Table 2: Breakdown of questions across 3 dimensions

and associated metadata were subsequently reviewed by other annotators to verify their validity and correct minor issues, such as typos or associated metadata.

Hierarchical Column/Row Header Identification. To identify column and row headers of tables, we leveraged Table Understanding technology incorporated in IBM Watson Discovery[3]. Table Understanding allows among others identifying for each body (i.e., non-header) cell, the set of column headers and row headers that describe the cell[4]. Table Understanding also supports column and row header hierarchies, as described above. The identified header hierarchies are included as part of the dataset so that they can be leveraged by Table QA models.

Dataset Statistics. The resulting test dataset consists of 515 questions generated out of 116 tables. These tables were chosen from 13 10-K forms of the 5 considered airlines filed for years between 2017 and 2019. Table 2 shows the breakdown of questions along the three aforementioned dimensions.

4 Experimental Evaluation

To analyze the effect of AIT-QA's domain-specific complex tables to existing Table QA approaches, we next provide a comprehensive evaluation of state-of-the-art Table QA models on it.

4.1 Experimental Setting

We evaluate three Table QA systems - **RCI** (Glass et al., 2021), **TaBERT** (Yin et al., 2020), and **TaPaS** (Herzig et al., 2020) - selected as representing SOTA Table QA approaches of different architectures.

TaBERT employs an encoder-decoder approach, utilizing a BERT (Devlin et al., 2019) encoder and LSTM decoder, which generates intermediate logical forms which - when executed over tables - yield the answer (Liang et al., 2017). In contrast, TaPaS (Herzig et al., 2020) and RCI (Glass et al., 2021) both treat Table QA as a classification problem. However, TaPas considers tables as a whole, while RCI splits them into rows and columns and carries out inference on them separately. In all three systems, tables and questions are encoded using transformers (Vaswani et al., 2017).

To test whether high Table QA performance reported on open-domain tables translates to the domain-specific AIT-QA dataset, all three Table QA models are pre-trained on the larger WikiSQL (Zhong et al., 2017) train split and tested on AIT-QA without any hyper-parameter tuning. To set up the baselines, we use the source code and instructions of the respective authors (see Appendix B).

4.2 Transforming Table Structures

Existing table QA models are based on open-domain web tables. They assume that the input tables contain flat column headers (i.e., a single row of column headers) and no row headers. Therefore, none of the existing baselines are built for handling complex column or row header hierarchies seen in AIT-QA. Thus, we experiment with table transformation operations to maximize these baselines' performance on AIT-QA.

Base transformations are first performed on AIT-QA tables to render the tables compatible to the models: (1) Row headers are added as the first column of the table as regular body cells. We use the dummy text 'header' as the column header of the new column. (2) Header hierarchies are flattened by concatenating parent header text with children text. Note that these transformations are designed to help the baseline models perform better than if we ran them on the raw table. For instance, when converting the table of Figure 1, the cell on the left of the blue cell will contain the concatenated row header hierarchy (i.e., 'Mainline passenger revenue miles (millions)'). This should help the models (which are not built to recognize row header hierarchies) perform better on AIT-QA.

Transposing tables. However, after running the models, we observed that there was further room for improvement. In particular, we observed that in many tables in AIT-QA, row headers contain more information than column headers. For example, in Figure 1, row headers contain the metric names, which are arguably more descriptive than the column headers containing the year information. Based on this intuition, we experimented with transposing the headers, so that row headers become column headers (which

[3] https://www.ibm.com/cloud/watson-discovery

[4] https://cloud.ibm.com/docs/discovery-data?topic=discovery-data-understanding_tables

Version	TaBERT	TaPaS	RCI
Base	33.20	49.32	40.58
All T	33.39	43.88	48.54
Partial T	33.98	46.80	51.84

(a) Accuracy of Table QA on different transformations of the tables in AIT-QA (**Base** = No transpose, **All T** = All transpose, **Partial T** = Partial Transpose).

Data subset	TaBERT	TaPaS	RCI
Overall accuracy	33.98	49.32	51.84
KPI-driven	41.37	48.26	60.00
Table-driven	31.08	50.0	48.64
Row header hierarchy	21.92	47.26	45.89
No row header hierarchy	38.75	50.39	54.20

(b) Accuracy of Table QA models on slices of AIT-QA

Table 3: Accuracy of Table QA models on AIT-QA

the models are trained to pay more attention to) and vice versa. During this process body cells are appropriately transposed as well. This led to three versions of AIT-QA data: (1) *Base:* without transposing tables, (2) *All transpose:* With all tables transposed, and (3) *Partial transpose:* Transposing tables that have more characters in row headers than column headers. Table 3a depicts the accuracy of baseline models on each dataset version. Interestingly, RCI and TaBERT benefit from transposing, while the performance of TaPas declines. For our analysis below, we pick for each model the version of the data that yields the highest performance.

4.3 Analyzing Baseline Performance on AIT-QA's Dimensions

The first row of Table 3b shows the overall accuracy of the baselines on AIT-QA. Performance is relatively low ranging from 34% (for TaBERT) to 49% / 52% (for TaPaS / RCI, respectively). For reference, the accuracy of the models on the dev split of the WikiSQL dataset is significantly higher, ranging from 70.5% (for TaBERT) to 89.2% / 89.8% (for TaPaS / RCI, respectively). This verifies our intuition that open-domain datasets do not reflect the intricacies of domain-specific use cases, which was the main motivation for the creation of AIT-QA.

To gain further insights on how domain vocabulary, table structure, and question phrasing affect the performance of Table QA models, we next evaluate the models on the three dimensions of our dataset:

KPI-driven vs Table-driven. Table 3b shows the performance of the baselines on KPI-driven vs Table-driven questions. As shown, accuracy is always higher for the former than the latter. While identifying the reason behind this is beyond the scope of this paper (being related to the promising area of explainability of AI models), there are two factors that may be potentially contributing: First, KPIs are limited in number and often easily differentiable from other terms in the tables, which may help baselines identify the right answer. Second, KPI-driven questions were formed by having a KPI in mind and searching for the corresponding term in the document. In contrast, table-driven questions were formed by looking at a table and trying to form a question. As a result, it is much more common to find distorted utterances of row/column headers in table-driven questions, making it harder for the baseline to identify the correct answer.

Paraphrase	TaBERT	TaPaS	RCI
All correct	25.00	38.88	33.33
Any correct	29.17	30.55	34.72
All wrong	45.83	30.55	31.94

Table 4: Percentage of paraphrased question sets that are (a) all correctly answered, (b) at least one correctly and another one incorrectly answered, and (c) all incorrectly answered by each baseline

Row Header Hierarchy vs No Row Header Hierarchy. One of the key challenges associated with AIT-QA are row/column header hierarchies. While we tried to help the baselines (which have not built with complex header structures in mind) deal with hierarchies (see table transformations in Section 4.2), this implicit treatment of headers has two important limitations: (1) the explicit hierarchical information is lost and (2) in some cases, transformations may add noise into a row/column. Therefore, it is not surprising that questions that depend on row header hierarchies negatively affect the performance of all baselines and cause an average drop of ~10 percentage points (see Table 3b). This indicates that additional work is needed to make Table QA models better leverage row header hierarchies.

Paraphrasing. Paraphrasing is an important aspect of Table QA systems. AIT-QA contains 76 paraphrased questions, which can be grouped into 72 paraphrased question sets (i.e., sets that include the original questions and all its paraphrases). To study the effect

of paraphrasing, we computed for each approach the percentage of question sets for which (1) all questions were answered correctly (referred to as *All Correct*), (2) at least one question was answered correctly and another incorrectly (*Any Correct*), and (3) all questions in the set were answered incorrectly (*All Wrong*).

Table 4 shows the resulting percentages for all baselines. For instance, for RCI the percentages for (1) / (2) / (3) are ~33% / 35% / 32%, respectively. Out of the three categories, especially interesting is the 2nd category, as it represents questions that are supported when phrased in one way but not supported when phrased in a different way. With almost 30-35% of question sets in this category, Table QA systems seem to be very sensitive to question phrasing; another area that would benefit from additional work.

5 Conclusion

Table QA systems have shown high performance on existing Wikipedia-based datasets with simple tables. To understand whether they perform as well on domain-specific datasets commonly encountered in the industry, we created AIT-QA; the first Table QA dataset that explicitly captures domain-specific tables with complex structure, including column and row header hierarchies. Our experiments show the deficiency of SOTA Table QA approaches in this setting. We hope that this work and dataset encourage the community to consider new Table QA approaches that can support such complexity, so that Table QA methods can more effectively support domain-specific scientific and business use cases.

References

Doug Burdick, Marina Danilevsky, Alexandre V. Evfimievski, Yannis Katsis, and Nancy Xin Ru Wang. 2020. Table extraction and understanding for scientific and enterprise applications. *Proc. VLDB Endow.*, 13(12):3433–3436.

Michael J. Cafarella, Alon Halevy, and Nodira Khoussainova. 2009. Data integration for the relational web. *Proc. VLDB Endow.*, 2(1):1090–1101.

Mustafa Canim, Cristina Cornelio, Arun Iyengar, Ryan Musa, and Mariano Rodriguez-Muro. 2019. Schemaless queries over document tables with dependencies. *CoRR*.

Wenhu Chen, Ming wei Chang, Eva Schlinger, William Wang, and William Cohen. 2021. Open question answering over tables and text. *Proceedings of ICLR 2021*.

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.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Michael Glass, Mustafa Canim, Alfio Gliozzo, Saneem Chemmengath, Vishwajeet Kumar, Rishav Chakravarti, Avi Sil, Feifei Pan, Samarth Bharadwaj, and Nicolas Rodolfo Fauceglia. 2021. Capturing row and column semantics in transformer based question answering over tables. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1212–1224, Online. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Sujay Kumar Jauhar, Peter Turney, and Eduard Hovy. 2016. Tabmcq: A dataset of general knowledge tables and multiple-choice questions.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33, Vancouver, Canada. Association for Computational Linguistics.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Benjamin Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, and Dragomir Radev. 2022. FeTaQA: free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10(0).

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 (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Roee Shraga, Haggai Roitman, Guy Feigenblat, and Mustafa Canim. 2020a. *Ad Hoc Table Retrieval Using Intrinsic and Extrinsic Similarities*, page 2479–2485. Association for Computing Machinery, New York, NY, USA.

Roee Shraga, Haggai Roitman, Guy Feigenblat, and Mustafa Canim. 2020b. Web table retrieval using multimodal deep learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1399–1408, New York, NY, USA. Association for Computing Machinery.

Yibo Sun, Zhao Yan, Duyu Tang, Nan Duan, and Bing Qin. 2019. Content-based table retrieval for web queries. *Neurocomputing*, 349:183–189.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Shuo Zhang and Krisztian Balog. 2018. Ad hoc table retrieval using semantic similarity. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1553–1562, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

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, Online. Association for Computational Linguistics.

A Annotation Tool

To generate the question-answer pairs, annotators employed a custom-built Table QA annotation tool, a screenshot of which is shown in Figure 2. Using the tool, annotators can create question-answer pairs by (a) entering the question and its classification along the three dimensions of the dataset on the top of the screen and (b) selecting the table cell that contains the answer. As an annotator specifies questions, these are displayed in blue boxes above the tables containing the corresponding answers to allow for easier inspection. Finally, a counter of annotations is provided on the left side of the screen to allow annotators to keep track of their progress.

B Implementation Details for Baseline Table QA Models

To aid in reproducability, we next provide details on the codebases and processes used to create the state-of-the-art Table QA models employed in the experiments described in Section 4. As part of the experiments, we evaluate three Table QA systems: RCI (Glass et al., 2021), TaBERT (Yin et al., 2020), and TaPaS (Herzig et al., 2020), all pre-trained on the larger WikiSQL[5] (Zhong et al., 2017) train split. In all cases we use the original source code released by the respective authors, pretrained weights along with details in their papers, for setting up all baseline models. In particular:

- For TaBERT (Yin et al., 2020), we use the pre-trained BERT released on the official GitHub repository[6] with semantic parser MAPO[7]. TaBERT is trained for 10 epochs on 4 Nvidia Tesla v100s with a batchsize of 10, number of explore samples as 10 and all other hyperparameters kept exactly the same as (Yin et al., 2020).

- For RCI (Glass et al., 2021), we use the code released with the paper[8] to train the model for 2 epochs on 2 Tesla v100s, with learning rate 2.5e-5 and batch size 128. All the hyperparameters are kept the same as described in Section A.2 of (Glass et al., 2021).

- For TaPaS (Herzig et al., 2020), we use the model[9] trained on the WikiSQL dataset from the official GitHub repository[10].

C Ethical Considerations

The main goal of this work is to (a) inform the research community of the challenges that state-of-the-art Table QA approaches face when applied to domain-specific settings, and (b) provide resources (including the dataset and the insights in Section 4) to help address these challenges. As a result, we believe that this work has the potential to bring Table QA research closer to the real needs of users interested in getting answers to questions over tables found in domain-specific business and scientific documents, as commonly encountered in industrial settings.

While using the AIT-QA dataset for the above purpose, it is important to understand that the dataset represents a single domain/use case and may not be entirely representative of other domains/use cases. Based on our experience, the complex tabular structures encoded in AIT-QA (incl. column and row hierachies) are representative of structures found in many other scientific and business documents (e.g., medical papers with tables containing the results of clinical trials, licensing agreements with tables describing details of the agreement, etc.). However, other domains may have their own peculiarities (e.g., different vocabulary or specific table templates). As a result, even though AIT-QA can be used to get a first indication of the performance of a Table QA system in a domain-specific setting, we encourage the research community to also look at additional domains/use cases, as each one may have its own unique characteristics and associated challenges.

Including several domain-specific datasets in the evaluation of Table QA systems can also help ensure that when we design such systems, we take into account the needs of diverse sets of potential users and avoid introducing unwanted bias. As a concrete example, we believe that more work should be done in multi-lingual settings, as most Table QA datasets (including AIT-QA) focus on documents and questions written in English.

[5] https://github.com/salesforce/WikiSQL

[6] https://github.com/facebookresearch/TaBERT

[7] Source code available at https://github.com/pcyin/pytorch_neural_symbolic_machines

[8] https://github.com/IBM/row-column-intersection/

[9] https://storage.googleapis.com/tapas_models/2020_08_05/tapas_wikisql_sqa_masklm_large_reset.zip

[10] https://github.com/google-research/tapas

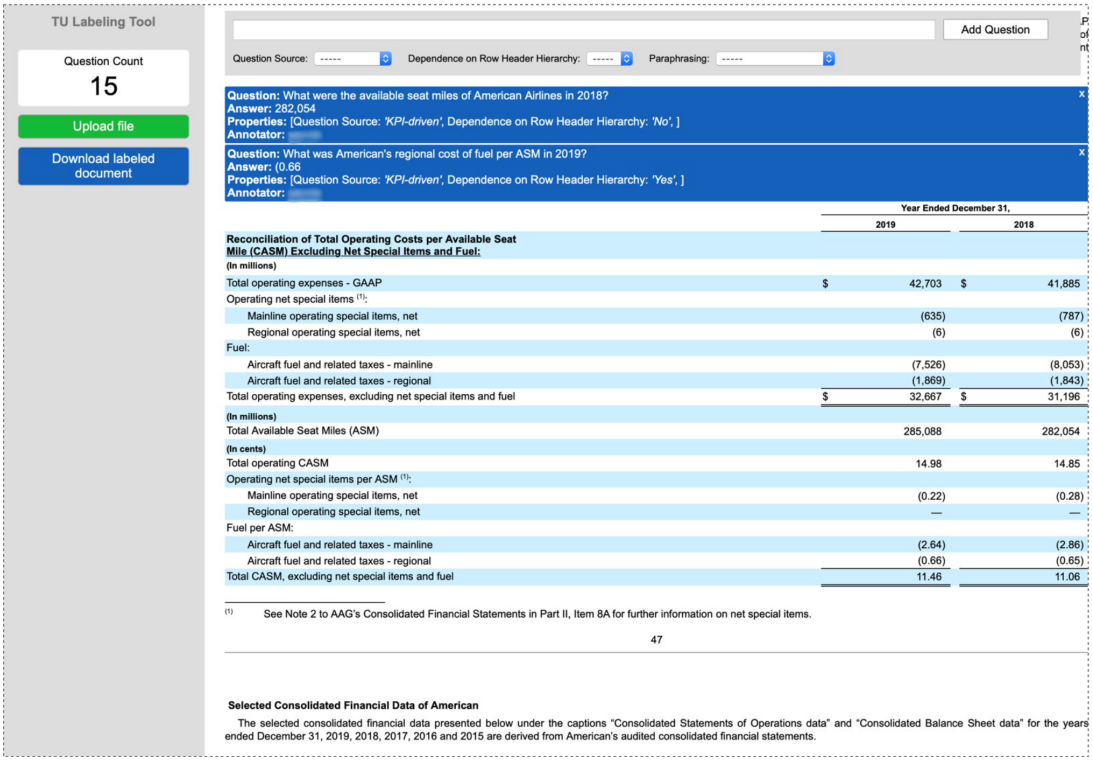

Figure 2: Screenshot of annotation tool used to create AIT-QA