Investigating Free-Form Table Question Answering with FeTaQA

Jiho Kim
kim999@wisc.edu

Sandeep Singh
ssingh272@wisc.edu

John Daniels
jrdaniels2@wisc.edu

1 Overview

Question answering (QA) is a Natural Language Processing (NLP) task focused on providing answers to questions posed by humans. The task combines elements of information retrieval, usually querying a structured database or knowledge base, as well as converting user-provided natural language queries to a form that can be used to retrieve the correct answer. One focus within QA is table question answering, where the questions posed, and answers retrieved, focus on the content of data organized into a table. For some larger tables, this can be used to answer questions that would otherwise require a significant amount of manual analysis and sorting through table cells.

Existing table QA datasets frequently involve simpler questions and short-form answers, often with little explanation. Real-world questions posed by humans are often more complex and require more entity retrieval and cross-referencing to answer, and those answers tend to be longer and include an explanation. One dataset designed with these real-world considerations in mind is the Free-form Table Question Answering dataset, or FeTaQA (Nan et al., 2021). Unlike generative QA datasets which contain the answers in a snippet of text, the answers to the questions in FeTaQA require forming higher-level relations between information retrieved from multiple points in the source table to produce the intended answer.

This project aims to explore the table question answering task as well as the FeTaQA dataset in particular. Our initial goal will be to reimplement the end-to-end model outlined in the FeTaQA paper, and compare our results to those obtained by the original authors. Our analysis of the results will provide us a better idea of which questions and input tables cause the benchmark models to struggle the most. Using these insights, we will plan further methods and experiments on the dataset with the goal of producing an improved table QA model that is capable of handling the challenging questions provided and generating free-form, intelligible, and correct responses to those questions. In addition to improving model performance on the dataset, we plan to examine which models perform best on questions that are incorrectly answered by our benchmark models. By comparing the performance of additional models with the type(s) of questions best answered by each model, we are interested in learning if there exists an ensemble method that can achieve better overall performance over any individual model.

2 Related Work

Yamada et al. (2020) proposed a model for entity representation with applications in question-answering. Their model uses a bidirectional transformer trained using a masked language model (MLM) pretraining task similar to that of BERT to optimize both entity representation and MLM tasks. Further, the model used an entity-aware self-attention mechanism to use both words and entities when calculating attention scores. Their model achieved good performance on both cloze-style and extractive question answering (Yamada et al., 2020). The questions included in the FeTaQA dataset frequently require identifying the high-level relations between entities in the input data to produce acceptable answers. By leveraging the features in this model, we may improve upon the accuracy of our benchmark model responses.

The table question-answering system proposed by Kumar et al. (2021) focused on a hybrid context, where answering questions may require cross-referencing table contents with free text from elsewhere in the document. Their approach split the question answering process into separate row selection and answer generation problems. Row retrieval is performed using a BERT-based model to classify the sequences as either containing or not containing the answer text, while adjusting for the situation where multiple rows may contain the correct information. Meanwhile, the answer generation is done by simulating a joint selection of the retrieved table rows and the text spans containing the answer text (adjusted for cases of "noisy" instances) to select the best answer (Kumar et al., 2021). The HybridQA dataset used in the paper bears some similarity to the FeTaQA dataset, as the two both focus on table QA using text and table data sourced from Wikipedia. However, the HybridQA dataset specifically tests short-form answers (an average of 2.1 words per answer), while the FeTaQA dataset focuses on longer, free-form text responses to more complex questions based entirely on the provided table.

Kim et al. (2020) explored visual question answering by initially surveying human-generated questions and explanations derived from visual charts, then using those results to guide the creation of a visual question-answering system. Their approach consisted of using a Vega-Lite representation of the input charts to convert a visual question to a non-visual question, and then using Sempre, a relational data table question-answering system, to compute the result. Using the Sempre result, the system then generates a visual explanation of how it derived the answer. Their results showed a marked improvement over using Sempre alone to answer visual questions, and even a slight improvement when used to answer non-visual questions (Kim et al., 2020). This model bears some conceptual similarity to the end-to-end benchmark model used to test the FeTaQA dataset, where a semi-structured table is used to generate the answer. However, the end-to-end model treats the table as input, appended to the input question rather than using the table to convert a visual question to a non-visual one.

Text-to-Text Transfer Transformer (T5) is a Transformer-based model that treats all text-based language problems as a text-to-text task. As a result, it can use the same model, loss function, training procedure, and hyperparameters for a variety of tasks, from summarization to machine translation and more. Developed with the goal to demonstrate general language learning, the model was tested on the GLUE and SuperGLUE text classification benchmarks, which contain a set of tasks to test sentiment analysis, coreference resolution, question answering, paraphrasing, and more (Raffel et al., 2019). When benchmarking FeTaQA, the authors included an end-to-end model which treated the question answering task as a sequence-to-sequence learning problem, appending a linearized table to the question as the initial sequence, and using a T5-family model to produce the answer sequence. This end-to-end benchmark achieved the highest performance in most of the evaluation metrics the authors examined (Nan et al., 2021).

FeTaQA addresses the need for a dataset that reflects the full challenge of generative table QA. The intent of the dataset is to challenge question-answering models with questions that require retrieving and comparing multiple pieces of data from a moderately large semi-structured table, and to produce answers that incorporate the retrieved information in a coherent and fluent sentence. The data was originally collected from ToTTo (Kale and Rastogi, 2020), a table-to-text dataset collected from Wikipedia. This data was then sampled to include only tables that met size requirements before being used to generate appropriate questions by using workers on Amazon Mechanical Turk. The end result is a complex dataset meant to provide a challenging table QA task using real-world tables and involved questions (Nan et al., 2021).

3 Reimplementation & Analysis

3.1 End-to-End Model - T5

As part of out reimplementation of the FeTaQA benchmarking, we used their end-to-end model and compared our results to theirs. The environment used to conduct our tests was the Google Cloud Platform, on a VM instance using a single Nvidia Tesla K80 GPU, which was chosen as it is the same type of GPU used in the original paper (though they used 4 such GPUs to train their t5-small and t5-base models, and 8 to train their t5-large model). Due to hardware limitations, we were unable to load the t5-large model that the original authors used into GPU memory, so we only recreated the t5-small and t5-base results.

Our models showed similar results to those of the original authors when tested using BERTscore and sacreBLEU metrics, with the t5-small model producing a BERTscore precision of .93 and a sacreBLEU score of 20.19, and the t5-base model producing a precision of .94 and a sacreBLEU score of 26.42. These are slightly lower than the results for the equivalent models from the original paper, which may be attributable to the amount of training time invested in the original models. As mentioned, due to time and hardware constraints we were only able to train the t5-base model for 10 epochs and the t5-small model for 30 epochs, compared to the 80 training epochs performed by the original authors. Additionally, our results are notably lower than those achieved by the t5-large model in the original paper, which we were unable to test due to the size of the model being too large for the hardware we had available for training.

Model	BERTscore	sacreBLEU
t5-small – Ours	.93	20.19
t5-base – Ours	.94	26.42
t5-small – Original	.94	21.60
t5-base – Original	.96	28.14

Assessment Method Used	Characteristics of Incorrectly Answered Questions
BERTscore	• Listing Elements • Repeating Incorrect Elements
sacreBLEU	• Summarization Tasks
Manual Error Analysis	• Answering multiple questions • Mismatch of Nomenclature • Inferring Numerical Operations from Attribute Names • Ambiguous Questions

Figure 1: The characteristics of questions that were incorrectly answered by the benchmark system. Different traits were found with respect to the assessment technique used.

3.2 End-to-End Model - BART

To compare the original implementation with another model, we selected BART (seq2seq conditional generative model) for our End-to-End model setup. BART is an encoder-encoder transformer model with a bidirectional encoder just like BERT and an auto-regressive decoder. It is pre-trained by, first corrupting the text with an arbitrary noise and then learning a model to reconstruct the original text. BART is effective particularly when fine-tuned for summarization and even for question-answering. The checkpoint that we took for our experiment was trained on CNN Daily Mail, a huge collection of text-summary pairs. We fine-tuned the pre-trained model on our FeTaQA dataset. The environment used to conduct this experiment was the same as our base End-to-end model re-implementation setup. We used the same tokenizer as of T5 to maintain the consistency of inputs in both the models. We trained the BART model for 10 epochs to have a comparable metrics with our previous experiment. Due to time and hardware constraints, our model is still under training phase. We will update the results soon.

Model	BERTscore	sacreBLEU
t5-small – Ours	.93	20.19
t5-base – Ours	.94	26.42
BART – Ours		

3.3 Qualitative Analysis

We performed a qualitative analysis on the system's predictions by comparing them to the ground truth. The analysis portrays a tangible picture of how the system behaves in certain situations and provides insight into what problems can be addressed in future iterations of the system.

First, we analyzed the questions that had low assessment metrics suggested by the original paper, including BERTscore (Zhang et al., 2019). We characterized the unintended behaviors of the system into the following:

Listing Elements: The system sometimes listed many elements without providing instead a summary of the elements. For example, "Which ships were sunk by U-511?" was answered with "U-511 sank Esso Aruba, Rotterdam, San Fabian, William Wilberforce, Sebastien Cermeno, Samuel Heintzelman, and Samuel Heintzelman." Though technically correct, the prediction is not as fluent as the ground truth "German submarine U-511 had sunk five ships totalling 41,373 gross register tons (GRT) and damaged one of 8,773 GRT." which gives more contextual information about the weight of the ships, and a more fluent sentence by aggregating the elements by counting them. Similarly, for the question "What is Hulchul?" the system produced "Hulchul is a Tamil film starring Rani Bhagwan, Rani Bhagwan, Krishna Kumari, Yashodhara Katju, Pratima Devi, Cuckoo, Gajanan Jagirdar, Ajit, Nigar Sultan", making the sentence unnecessarily prosaic by enumerating the actors, compared to the ground truth "Hulchul is a 1951 Hindi film directed by S K Ojha."

Repeating Incorrect Elements: In some predictions with low BERTscore, data values irrelevant to the questions were repeated. For example, 'Which films did Raghava Lawrence act in 2000?' was answered with a list that contains the film “Style”, which is in fact not a part of the correct answer. Likewise, for “How many aircraft does Japan Airlines currently operate and which manufacturers make them?”, the system answered “Japan Airlines operates a fleet of 737, Boeing, 737, 777, 777, 777, 777, 777, 777, 777, 777, 777, 777, 777, 777, 777, 777, 777.” Though it is difficult to deduce its exact cause, we did notice that the linearized table representations that were provided with the two questions above both contained empty cells and merged cells. We speculate that the system struggles to handle these unconventional tables.

Second, predictions that scored the lowest with sacreBLEU (Post, 2018) metric were analyzed. The analysis drew out issues different from those elicited from analyzing with BERTscore.

Summarization Tasks: The system struggled with summarization tasks, which involve not only retrieving values from the table, but also combining multiple values to form a coherent, highly informative sentence. Predictions with low sacreBLEU scores often contained correct information, but were not contextually informative enough compared to the ground truth answers. For example, “What happened during the release of The Rake’s second album in 2007?” was answered with “The Rake’s second album Ten New Messages was released in 2007.”, failing to add anything new to the information already insinuated in the question. Sometimes, the high complexity of ground truth answers caused the low score. For instance, "Which candidates were Fauntroy’s top two competitors in the 1978 United States House of Representatives election in D.C. and what percentage of the vote did they earn, respectively?" was answered with "Fauntroy won 79.59 percent of the vote against Champion’s 12.02 percent.", which is indeed correct, but insufficient compared to the ground truth(“Fauntroy was opposed in 1978 United States House of Representatives election by Republican challenger Jackson R. Champion and Statehood Party candidate Gregory Rowe who received 12.02% and 4.04%, respectively”).

Third, we created a Python script that makes it easy to compare predictions and ground truth answers, and manually went over the questions to see if any other interesting themes emerged.

Answering multiple questions: Some questions that comprise more than one sub-questions were not answered completely. For example, “When did the 1966 Tour de France take place, how many stages did it consist of, and what was the total distance covered by the event?” is composed of three different questions(when, how many, and what). The prediction (“1966 Tour de France consisted of 21 stages covering a total of 2,090 km (2,050 mi).”) answers two of them, but overlooks one.

Mismatch of Nomenclature: The question can contain words or phrases that do not exactly match the nomenclature used in the table. For example, to correctly answer “How many games did Shay Given play during the *2014-2015* Premier League season and for which club?” the system needs to correspond “2014-2015” in the question to “2014-15” in the table. Failing to do so, the system generated an incorrect answer that refers to a different time period. The system generally performed well when there was a mismatch between two English words, meaning that numbers, when used as a categorical variable, cause problems. This makes sense as semantic similarity is more difficult to check between numbers/symbols than between words.

Inferring Numerical Operations from Attribute Names: The system could understand the basic semantics of each word and phrase in the question as well as the table. However, in cases where data operations have to be deduced from attribute names, the system made mistakes. For instance, “How many Class NG G11 Garratt locomotives did the South African Railways place between 1919 and 1925?” requires the system to locate the said locomotives, retrieve the number in the “Units” column, and add them up. This requires understanding of what the column “Units” mean. However, the system just answered with the number of rows that contained “NG G11”, without adding the numbers in the “Units” column.

Ambiguous Questions: There were a number of ambiguous questions that can be interpreted in different ways, thus allowing for more than one correct answers. To be specific, which table cells to refer to is unclear from the question, depending on the interpretation. For example, the ground truth answer for “How many goals did Wright score with Walsall after returning to the club?” is “Wright returned to Walsall and scored 38 goals in 152 league games with the club.” However, 38 goals is the result of counting only the goals in the “League” column. Referring instead to the “Total” column in the table gives the answer “43 goals”, which is exactly what our system predicted. One could easily make the case that the answer to the question is “43”, not “38”. This shows that the requirement for the answer is loosely defined, which can negatively influence the training.

4 Potential Methods

In addition to the end-to-end model reimplemented in this paper, the original FeTaQA authors implemented a pipeline model composed of a weakly supervised table semantic parser combined with a data-to-text model. While the pipeline model demonstrated decreased accuracy along all metrics when compared to the end-to-end model, it was still capable of producing fluent responses to input questions. While our original reimplementation only focused on the most successful model, further investigation into the FeTaQA dataset and the best ways to improve table QA performance could benefit from examining how well the pipeline model performs and on which questions it excels – or struggles.

We can also expect improvement from replacing or refining each module in the pipeline model. The original pipeline model mentioned in the paper incorporates TAPAS (Herzig et al., 2020), which is a semantic parser for for table question answering. We can use other table parsers such as Sempre (Berant et al., 2013), RCI (Glass et al., 2021), and TaBERT (Yin et al., 2020). This expectation is reasonable as it has been shown in prior work that other table parsers can outperform TAPAS in certain tasks (Katsis et al., 2021).

The pipeline model uses the cell highlight data from ToTTo (Kale and Rastogi, 2020) as a weak supervision, to fine-tune TAPAS. However, TAPAS is pre-trained on questions that are much simpler than the FeTaQA’s annotated questions. Limitations in performance can arise from this pipeline. Furthermore, the denotations provide information about which cells to refer to when answering a given question. Nonetheless, there is no hierarchy between the cells. In other words, all highlighted cells are treated equally. This can be adequate enough for questions that ask for simple value retrieval or an aggregate such as *COUNT* and *SUM*, but for more complex questions, more informative denotations are wanted. Thus, an automated way to rank the highlighted cells according to its significance can help improve the system’s performance.

In the process of analyzing the system’s predictions and the common inaccuracies when answering questions, we discovered that some questions or answers cannot be understood clearly. In addition, the denotations used for the weak supervision, or the highlighted cells, were sometimes incorrect upon careful investigation. As the FeTaQA dataset was produced and reviewed by human annotators, such noises in the data are understandable. By identifying and either correcting or removing these errors, we may be able to improve the dataset itself. Further analysis could then be performed by fine-tuning our models on the improved dataset, and comparing the performance of the two setups to find any examples of questions that produce either improved or worsened responses.

5 Experiments

To gain better insight into the performance of various models on the FeTaQA dataset, we would reimplement the pipeline model described in the original paper. We do not anticipate the performance of the pipeline model exceeding that of the end-to-end model, but we may find that the pipeline model performs better on a a different set of questions than that of the end-to-end model. By utilizing an ensemble composed of multiple models with different strengths, we hope to improve on the peak performance found in the original paper. In addition to the pipeline model, we plan to experiment with additional models to determine if an effective ensemble can be produced to obtain better predictive performance when compared to any single model.

While our experimental performance using the T5-based end-to-end model was comparable to the results achieved by the original authors, it fell slightly short when using comparable models. One likely factor to explain the decreased performance metrics is that, due to time constraints, we were not able to devote as much time to training the models on the FeTaQA dataset. Continuing to train our end-to-end model implementations, ideally for the full 80 training epochs used by the original authors, should provide peak accuracy for our model (and any ensemble models that may make use of the model).

Through the qualitative analysis, we revealed the characteristics of questions that were incorrectly answered by the system. We can classify the questions in the dataset by leveraging this set of characteristics. The classification will be done

Week	Tasks
3/20 – 3/26	• Find a good paper for re-implementation • Search prior work on table question answering
3/27 – 4/2	• Set up the environment using Google Cloud Platform • Understand the model conceptually and analyze the code • Start reimplementing the end-to-end model, as described in the original paper
4/3 – 4/8	• Finish the reimplementation of the end-to-end model • Train the model and retrieve the predictions • Analyze the predictions • Try incorporating different models (BART)
4/10 – 4/16	• Start implementation of the pipeline model, as described in the original paper • Preprocess (Categorize) the questions to expedite future prediction analysis
4/17 – 4/23	• Finish implementation of the pipeline model • Run the pipeline model and analyze its predictions, and compare it to end-to-end model
4/24 – 4/30	• Experiment with different setups for the pipeline model & end-to-end model • Incorporate other models and analyze their performance
5/2 – 5/6	• Finalize the ensemble model • Retrieve improved result over the original work • Write up the final report

Figure 2: Timeline that shows the tasks to be completed each week until the deadline of the final project.

first manually by a researcher for a small set of the questions that were not answered correctly. Then, after testing the dataset on different proposed models, we can analyze the predictions based on this classification to see if there exists any relationship between a category and a model. This will allow for a more granular view of the system when assessing different models before creating an ensemble model.

6 Plan of Activities

Figure 2 shows the overall agenda for this project. The workload will be divided between the three members through a weekly virtual meeting. In addition, Microsoft Teams will be used to communicate between team members. All artifacts will be uploaded to the project Github repository.

We will collaborate on writing the final report using Overleaf. The report will be gradually completed as each member writes about what they have done or have found while performing the tasks. In the last week, the report will be proofread by all members to ensure overall quality.

References

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

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. *arXiv preprint arXiv:2104.08303*.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks.

Yannis Katsis, Saneem Chemmengath, Vishwajeet Kumar, Samarth Bharadwaj, Mustafa Canim, Michael Glass, Alfio Gliozzo, Feifei Pan, Jaydeep Sen, Karthik Sankaranarayanan, et al. 2021. Ait-qa: Question answering dataset over complex tables in the airline industry. *arXiv preprint arXiv:2106.12944*.

Dae Hyun Kim, Enamul Hoque, and Maneesh Agrawala. 2020. *Answering Questions about Charts and Generating Visual Explanations*, page 1–13. Association for Computing Machinery, New York, NY, USA.

Vishwajeet Kumar, Saneem Chemmengath, Yash Gupta, Jaydeep Sen, Samarth Bharadwaj, and Soumen Chakrabarti. 2021. Multi-instance training for question answering across table and linked text.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Nick Schoelkopf, Riley Kong, Xiangru Tang, Murori Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, and Dragomir Radev. 2021. Fetaqa: Free-form table question answering.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.