1 Introduction

In the era of data-driven decision-making, tabular data plays a crucial role in facilitating data analysis, serving as a concise and structured representation of information (Kukich, 1983; Pasupat and Liang, 2015; Chen et al., 2020c; Zhu et al., 2021; Zhao et al., 2022a; Tang et al., 2023). People often consult tables to extract valuable insights and make informed decisions. For example, sales managers typically explore large tables with specific business questions to gain insights about clients and processes. Sports coaches will analyze performance tables containing various statistics to develop game strategies and make team adjustments. However, effectively accessing and comprehending the information contained within a large and complex table can be time-consuming for users (Hurst, 2000; Pasupat and Liang, 2015; Pujara et al., 2021; Nan et al., 2022a). Text generation systems that can accurately summarize a provided table according to users’ information needs have the potential to greatly enhance data analysis and expedite the process of obtaining data insights.

Existing work and datasets on table-to-text generation (Parikh et al., 2020; Chen et al., 2020a; Cheng et al., 2022b; Lebret et al., 2016; Moosavi et al., 2021; Suada et al., 2021) have mainly focused on converting tabular data into coherent statements, aiming to present the structured data in a human-readable format. However, these approaches have overlooked the fundamental goal of addressing...
To improve both text generation systems for QT-UMM, we propose REFACTOR. Given a user query, REFACTOR can retrieve and reason over query-relevant facts from the source table to generate multiple data insights in natural language sentences. Our results illustrate that directly concatenating the original input sequence with REFACTOR’s generation can bring effective improvements to state-of-the-art baseline systems.

We conclude our main contributions as follows:

- We propose a new query-focused table summarization task, and construct a large-scale benchmark, QTSUMM, comprising 7,111 query-summary pairs collected in real-world situations. Strict quality control measures are employed to ascertain the high quality of the dataset.
- We conduct a systematic study of state-of-the-art models on QTSUMM, and illustrate that they are still far behind expert performance, motivating future research on this new table-to-text task.
- We present REFACTOR for the efficient retrieval and reasoning of query-relevant facts from tables. It demonstrates significant enhancements pertaining to state-of-the-art text generation baselines.

2 Related Work

Table-to-Text Generation As illustrated in Table 1, existing work and datasets on table-to-text generation typically pose the problem as either a single-sentence generation task (Chen et al., 2020a; Parikh et al., 2020; Cheng et al., 2022b; Liu et al., 2022a), or a generic summarization task (Lebret et al., 2016; Moosavi et al., 2021; Suadaa et al., 2021). In the single-sentence generation task (Parikh et al., 2020; Chen et al., 2020a; Cheng et al., 2022b), the focus is on generating fluent and faithful descriptions using provided table regions as a control for text generation. Nevertheless, using table regions for controlling text generation does not align with real-world scenarios, where people refer to tabular data for information-seeking purposes. The generic table summarization tasks (Lebret et al., 2016; Moosavi et al., 2021; Suadaa et al., 2021) aim to create concise and informative summaries based on the content of a given domain-specific table (i.e., sports or scientific). In contrast, the tables in QTSUMM cover diverse topics. Furthermore, considering the numerous data points in the table, various users may be interested in different aspects for their own information-seeking purposes.
purposes, making it challenging to create a generic summary that encompasses all the salient information within the table. Therefore, in this paper, we propose and investigate a new task setting related to \textit{query-focused summarization}. FeTaQA (Nan et al., 2022b) is a table QA dataset that collects queries by rewriting ToTTo’s (Parikh et al., 2020) statements into questions and uses the same statements as the answers. In comparison with FeTaQA, the queries in QTSUMM were annotated under \textit{real-world scenarios}, making them more natural and better-reflecting users’ actual information needs.

\textbf{Reasoning Over Tabular Data} Enhancing the table reasoning capabilities of models is essential for a variety of tasks related to tables, such as table question answering (Pasupat and Liang, 2015; Iyyer et al., 2017; Zhong et al., 2018; Zhao et al., 2023d), table fact verification (Chen et al., 2020b), and table-to-text generation (Chen et al., 2020a; Cheng et al., 2022b). One prevalent approach is pre-training models with table-text joint reasoning data (Herzig et al., 2020; Liu et al., 2022b; Zhao et al., 2022b; Liu et al., 2022a; Jiang et al., 2022; Dong et al., 2022; Cheng et al., 2022a; Xie et al., 2022). Nevertheless, these models generate text in an end-to-end manner, resulting in reduced explainability and difficulties in handling more complex reasoning, such as arithmetic calculation. Therefore, we propose ReFACTOR, which can retrieve and generate query-relevant facts from tables as intermediate results for model input (Zhou et al., 2022; Zhao et al., 2023b), mitigating the \textit{implicit} reasoning processes of text generation models.

\textbf{Query-Focused Summarization} Initially formulated as a document summarization task, QFS aims to generate summaries from documents that are tailored to specific user queries (Dang, 2006). Despite its potential real-world applications, QFS remains a challenging task due to the lack of large-scale training data. Existing works have attempted to address this issue by leveraging distant NLP resources, including question answering (Xu and Lapata, 2020) and paraphrase identification (Su et al., 2020), and generic summarization (Xu and Lapata, 2022; Zhou et al., 2023). Recently, Zhong et al. (2021) adopted QFS for meeting summarization and proposed a human-annotated benchmark over meeting transcripts. Similar to text, effectively accessing and comprehending the information contained within a large and complex table can be time-consuming for users, while QFS remains unexplored in table-to-text generation. In this work, we extend QFS to this new modality for more effective information-seeking and decision-making purposes.

\section{Query-Focused Table Summarization}

\subsection{Problem Formulation}

We formally define the proposed query-focused table summarization task as follows. The input is a user query \( Q \), and a table \( T \). The table \( T = W \cup \{T_{i,j} \mid i \leq R_T, j \leq C_T \} \) has \( R_T \) rows and \( C_T \) columns, with \( W \) being the table title and \( T_{i,j} \) being the textual content in the \((i, j)\)-th cell. The task objective of QTSUMM is to generate a paragraph-long textual summary \( Y = (y_1, y_2, \ldots, y_n) \) given the user query \( Q \) and source table \( T \):

\[
Y = \arg\max_{\mathbf{y}} \prod_{i=1}^{n} P(y_i | y_{<i}, Q, T; \theta), \quad (1)
\]
where \( \theta \) denotes the parameters of a neural text generation model, and \( y_i \) denotes the \( i \)-th tokens in the generated summary.

### 3.2 Data Collection Principles

At a high level, the goal of the data collection process is to obtain high-quality user queries and corresponding paragraph-long summaries grounded on the tabular data. We outline our key criteria for designing a benchmark to thoroughly evaluate the table-to-text summarization capabilities of models. To achieve this, we first design three principles for annotating a good query-summary pair:

- **Comprehensiveness**: The tailored summary should provide enough details and analysis of the source table to respond to the user query, fulfilling user’s information need.

- **Attributability & Faithfulness**: The query should be answerable using only information from the source table. The summary should be grounded on the source table, and not contain any unfaithful or nonsensical text.

- **Fluency**: Both the user query and its corresponding table summary should be coherent and fluent.

### 3.3 QTSUMM Annotation Pipeline

To ensure that QTSUMM annotation fulfills the aforementioned principles, we carefully design an annotation pipeline consisting of following steps:

**Source Table Collection** QTSUMM uses tables from LOGICNLG (Chen et al., 2020a) and ToTTo (Parikh et al., 2020) datasets as source tables, as these tables are crawled from Wikipedia and cover diverse domains and topics. We filter out tables that are 1) too large or too small, 2) with only string-type columns, or 3) with hierarchical structures (e.g., containing more than one table header). Then we randomly sample 2,000 candidate tables from LOGICNLG and ToTTo, respectively, for the query-summary annotation.

**User Query Annotation** Given a table, the annotators are required to read its content, and determine whether the table is informative and intelligible to common web users. Then they were asked to come up with two or three queries, assuming they are users seeking certain information from the table. We require each query to be answerable using information only from the table. Moreover, as this work focuses on paragraph-long summaries as query responses, we avoid queries that can be answered in a short sentence (e.g., “Which country held the 2022 FIFA World Cup?”).

**Query-Focused Summary Annotation** Given a table and user query, we ask another annotator to use only information from the source table to write a paragraph-long summary that satisfies the user’s information need. We encourage annotators to produce sophisticated summaries that 1) contain as much information from the table as possible, and 2) involve more types of reasoning over multiple relevant table regions. To further encourage high quality annotations, we adopt the "two channel collection" design (Chen et al., 2020b), in which the annotators would be paid 60% more if their summaries are manually verified to exhibit adequate complexity. We also require the annotators to annotate the row indices of relevant table regions. To further encourage high quality annotations, we adopt the two channel collection design (Chen et al., 2020b), in which the annotators would be paid 60% more if their summaries are manually verified to exhibit adequate complexity. We also require the annotators to annotate the row indices of relevant table regions.

**Multi-Round Validation** We conduct a multi-round validation protocol to ensure that the annotated data fulfills the aforementioned annotation principles. We first assign query annotators to validate each summary against their corresponding query, and fix the mistakes if there are any. Then we check 1) whether a query-summary pair contains adequate information and complex aggregation by examining the length of the summary, and 2) whether the information in summary is essential in responding to the user query. We manually revise pairs that do not meet the above standard.

### 3.4 Annotation Quality Control

Table 2 describes the basic statistics of QTSUMM dataset. In addition to the multi-round validation, we carefully design several quality control approaches,
comprising expert annotation and numerous annotation de-biasing designs, to ensure the high quality of QTSumm annotations.

**Expert Annotators**  To help improve the annotation process, five experts with professional experience in the text summarization tasks are invited to conduct the internal annotation. They are asked to provide feedback regarding the task instructions and the user experience of the annotation interface, based on which we iteratively modify the annotation guideline and interface design. In the stage of external annotation, we enroll 17 graduate students majoring in STEM fields (10 females, and 7 males). We do not use the crowd-source annotation platform such as Mechanical Turk as our preliminary study indicates that annotators on MTurk fail to annotate high-quality query-summary data. Before starting the official annotation process, each annotator is given a two-hour training session to learn the annotation requirements and interface.

**Annotation De-biasing**  We observed several kinds of annotation bias during our internal annotation, and we proposed countermeasures as follows for annotation de-biasing:

*Source Table Diversity:* During internal annotation, we found that many tables in LOGIC-NLG have similar content. For example, there are around 200 tables describing the results of football games, with identical table headers. To ensure the diversity of source tables, we keep only one table for each unique table header.

*Query Diversity:* When annotating queries, annotators may prefer simpler ones, resulting in low query diversity. Therefore, we frequently monitor the diversity of queries for each annotator. Annotators are also encouraged to craft queries that are either creative or require complex reasoning in summarization, resulting in a doubled payment to compensate them for the extra time.

*Supporting Fact Position:* We found that annotators prefer to raise queries regarding the first few rows of each table. To deal with such bias regarding supporting fact positions, we randomly highlight certain rows for each table in the annotation interface. We require the annotators to write queries whose summaries should cover at least two rows of the highlighted regions.

We also report the human evaluation scores and inter-evaluator agreements over 200 sampled query-summary pairs. QTSumm has a high annotation quality and inter-annotator agreement (Table 3).

### 3.5 QTSumm Evaluation

We develop a comprehensive approach for evaluating QTSumm, incorporating both automated and human evaluation. We adopt following popular automated evaluation metrics:

**BLEU** (Papineni et al., 2002) computes the geometric average of the precision over output text’s n-grams. We used SacreBLEU (Post, 2018) that produces comparable and reproducible BLEU scores.

**ROUGE** (Lin and Hovy, 2003) measures the word overlap between the candidate and reference summaries. We reported F1 score for ROUGE-L (longest common subsequences).

**METEOR** (Banerjee and Lavie, 2005) is based on a generalized concept of unigram matching between the machine-produced translation and human-produced reference translations.

**BERTScore** (Zhang et al., 2020) computes the sim-
ilarity between the reference and generated summary using contextual word embeddings.

**TAPAS-Acc** (Herzig et al., 2020; Liu et al., 2022a) is a reference-free metric that uses TAPAS (Herzig et al., 2020) fine-tuned on the TabFact dataset (Chen et al., 2020b) as the backbone to evaluate the faithfulness of generation.

**AutoACU** (Liu et al., 2023a) is an interpretable and reference-based summarization evaluation system that exhibits better alignment with human judgments. The A2CU first extracts atomic content units (ACUs) from the generated summary and then evaluates them against reference. A3CU is an accelerated version of A2CU that directly computes the similarity between two text without extracting ACUs, but with the similar evaluation target. We use F1 score of A3CU for evaluation.

For **human evaluation**, the summaries from different models were evaluated by experts from three criteria (i.e., comprehensiveness, faithfulness, and fluency) that have been discussed in Section 3.2. Each summary was scored from 1 (worst) to 5 (best) for each criteria, with the final score averaged across different evaluators.

## 4 ReFACTOR

QTSUMM requires models to perform human-like reasoning in generating summaries that provide comprehensive and precise analysis of the source table to fulfill the user’s information need. However, existing end-to-end text generation models rely on error-prone **implicit** reasoning processes for generating text, leading to diminished explainability and challenges in addressing user queries that necessitate complex human-like reasoning (Zhou et al., 2022; Zhao et al., 2023b). To address this, we present ReFACTOR, to retrieve and reason over query-relevant information from tabular data to generate several NL data insights (i.e., facts) as explicit reasoning results. As shown in Figure 3, the generated facts are concatenated to the model input to mitigate the implicit reasoning issues, enhancing the comprehensiveness and faithfulness of generated summary. We next discuss the implementation of ReFACTOR.

### 4.1 Fact Generation

Given the user query and source table, ReFACTOR will generate several candidate facts by executing various forms of human-like reasoning over the table. Specifically, we define 6 types of table reasoning operations (e.g., numerical operation, counting, and conjunction) that are necessary for the QTSUMM task, as shown in Table 7 in the Appendix. For each reasoning operation, the fact generator (adopted from Zhao et al. (2022b)) takes a table and a query as input. It produces multiple facts based on the fact template. Each fact template includes several placeholders that need to be filled with information retrieved from the table. Specifically, column `col` and cell value `val` are indexed to specify the column and cell name, respectively. Some templates also regulate that the selected column and cell value must be date or number type. **OPERATOR** corresponds to operators that are instantiated according to the specific reasoning. And **CONDITION**: i can be 1) a cell value from the i-th column; or 2) a number/ temporal comparison statement if the i-th column is date or number type. After substituting all the placeholders in the provided template, the fact generator will programmatically return the `executed_results` and form one fact. Once facts for a {table, query} pair are collected from different fact generators, we pass them to the Fact Ranking process.
4.2 Fact Ranking
Given the query and source table, each fact generator will be utilized to generate several query-relevant facts, resulting in a large number of candidate facts in total. Therefore, we need to rank the generated facts to select the most relevant ones. We use the QA encoding model (Reimers and Gurevych, 2019) to obtain the embedding of the query and each generated fact. Then, we select the top-\(n\) generated facts with the highest cosine similarity to the query embedding. In practice, we assign \(n\) as \(\max\left(\sqrt{\text{rownum} \times \text{columnnum}}, 5\right)\), and ensure that the number of selected facts from each type of reasoning operation does not exceed 3. The selected facts, which are handy and readily available for end-to-end text generation systems, are then concatenated into the model input.

5 QTSUMM Experiments
5.1 Baseline Systems
We evaluate the following three types of state-of-the-art baseline systems\(^2\) on QTSUMM:

5.1.1 Text Generation Models
BART (Lewis et al., 2020) is a pre-trained denoising autoencoder with transformer-based architecture and shows effectiveness in NLG tasks.
T5 (Raffel et al., 2020) demonstrates effectiveness in NLG tasks by treating all NL problems as text-to-text tasks during pre-training stage.
Flan-T5 (Chung et al., 2022) enhances T5 by scaling instruction fine-tuning and demonstrates better human-like reasoning abilities than the T5.

5.1.2 Table-to-Text Generation Models
TAPEX (Liu et al., 2022b) continues pre-training the BART model by using a large-scale corpus of synthetic SQL query execution data. It shows better table understanding and reasoning abilities.
ReasTAP (Zhao et al., 2022b) enhances the table understanding and reasoning abilities of BART by pre-training on a synthetic Table QA corpus.
Omnitab (Jiang et al., 2022) uses the same backbone as TAPEX, and is further pre-trained on collected natural and synthetic Table QA examples.

5.1.3 Large Language Models
Llama-2\(^3\) (Touvron et al., 2023a,b) is an open-source large language model trained on large-scale and publicly available datasets.
Vicuna\(^4\) (Zheng et al., 2023) is tuned from Llama-1 with instruction-following data, exhibiting better instruction-following capabilities.
Mistral\(^5\) (Jiang et al., 2023a) is a 7–billion-parameter LLM that outperforms Llama-2-13B across most of popular evaluated benchmarks.
Lemur\(^6\) (Xu et al., 2023) is tuned from Llama-2 with instruction-following data, exhibiting competitive natural language and coding capabilities.
GPT (Brown et al., 2020; OpenAI, 2023) is a powerful large language model which is capable of generating human-like text and performing a wide range of NLP tasks in a few-shot setting.

5.2 Experimental Setup
The specifics of input data serialization and LLM prompting examples are discussed in Appendix A. All experiments were conducted on an 8 NVIDIA RTX A6000 48GB cluster. We selected the large version for all fine-tuned baseline models, whose weights are publicly available at HuggingFace. For each fine-tuning experiment, we ran 15 epochs with a batch size of 128. The best fine-tuning checkpoints were selected according to the validation loss. The experiments for open-sourced LLMs were conducted using vLLM framework (Kwon et al., 2023). We used gpt-3.5-turbo-0613 for GPT-3.5 and gpt-4-0613 for GPT-4 via the OpenAI APIs\(^7\). For LLM hyperparameter settings, we set temperature as 1.0, Top P as 1.0, and maximum output length as 256.

5.3 Main Results
We draw following conclusions based on the automated and human evaluation results (Table 4 & 6).

Importance of table structure understanding
Table-to-text generation models achieve better performance than their corresponding text-generation models at HuggingFace (https://huggingface.co/yale-nlp/{model_name}-finetuned-qtsum).

---
\(^2\)We released the model weights of evaluated fine-tuned models at HuggingFace (https://huggingface.co/meta-llama/llama-2-{size}b-chat-hf).
\(^3\)https://huggingface.co/meta-llama/llama-2-{size}b-chat-hf
\(^4\)We only evaluate Vicuna (https://huggingface.co/lmsys/vicuna-33b-v1.3) under zero- and one-shot settings, as some examples under the two-shot setting might exceed its maximum length limit.
\(^5\)mistralai/Mistral-7B-Instruct-v0.1
\(^6\)https://huggingface.co/OpenLemur/lemur-70b-chat-v1
\(^7\)https://openai.com/api/
<table>
<thead>
<tr>
<th>Type</th>
<th>Model</th>
<th>Backbone</th>
<th>BLEU</th>
<th>ROUGE-L</th>
<th>METEOR</th>
<th>BERTScore</th>
<th>TAPAS-Acc</th>
<th>A3CU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground Truth</td>
<td></td>
<td></td>
<td>67.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text Generation</td>
<td>T5-large</td>
<td>–</td>
<td>61.8</td>
<td>20.3</td>
<td>38.7</td>
<td>40.2</td>
<td>89.6</td>
<td>75.1</td>
</tr>
<tr>
<td>fine-tuning</td>
<td>Flan-T5-large</td>
<td>T5</td>
<td>74.1</td>
<td>19.9</td>
<td>39.8</td>
<td>42.5</td>
<td>89.8</td>
<td>83.9</td>
</tr>
<tr>
<td></td>
<td>BART-large</td>
<td>–</td>
<td>60.1</td>
<td>21.2</td>
<td>40.6</td>
<td>43.0</td>
<td>90.6</td>
<td>77.1</td>
</tr>
<tr>
<td>w: ReFACTOR</td>
<td>–</td>
<td></td>
<td>61.4</td>
<td>21.5 (+0.3)</td>
<td>41.0 (+0.4)</td>
<td>43.1 (+0.1)</td>
<td>90.1 (+0.5)</td>
<td>79.4 (+2.3)</td>
</tr>
<tr>
<td>Table-to-Text</td>
<td>ReaTAP</td>
<td>BART</td>
<td>61.4</td>
<td>22.5</td>
<td>41.9</td>
<td>44.3</td>
<td>90.8</td>
<td>80.6</td>
</tr>
<tr>
<td>fine-tuning</td>
<td>TAPEX</td>
<td>BART</td>
<td>70.1</td>
<td>23.1</td>
<td>42.1</td>
<td>45.6</td>
<td>90.6</td>
<td>87.8</td>
</tr>
<tr>
<td></td>
<td>OmniTab</td>
<td>BART</td>
<td>59.5</td>
<td>22.4</td>
<td></td>
<td></td>
<td></td>
<td>53.1</td>
</tr>
<tr>
<td>w: ReFACTOR</td>
<td>–</td>
<td></td>
<td>58.3</td>
<td>22.5 (+0.1)</td>
<td>42.2 (-0.2)</td>
<td>44.8 (+0.1)</td>
<td>90.7 (-0.3)</td>
<td>80.3 (+0.1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Model</th>
<th>Backbone</th>
<th>BLEU</th>
<th>ROUGE-L</th>
<th>METEOR</th>
<th>BERTScore</th>
<th>TAPAS-Acc</th>
<th>A3CU</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLM zero-shot</td>
<td>Llama-2:13B</td>
<td>–</td>
<td>64.3</td>
<td>14.6</td>
<td>25.5</td>
<td>30.9</td>
<td>86.8</td>
<td>76.6</td>
</tr>
<tr>
<td></td>
<td>Llama-2:7B</td>
<td>–</td>
<td>110.3</td>
<td>13.3</td>
<td>31.3</td>
<td>42.5</td>
<td>88.8</td>
<td>78.1</td>
</tr>
<tr>
<td></td>
<td>Mistral-7B</td>
<td>Llama-2</td>
<td>98.4</td>
<td>13.8</td>
<td>31.7</td>
<td>41.4</td>
<td>89.1</td>
<td>73.0</td>
</tr>
<tr>
<td>w: ReFACTOR</td>
<td>Llama-2</td>
<td>99.0 (+0.0)</td>
<td>31.4 (+0.3)</td>
<td>41.5 (+0.1)</td>
<td>88.7 (+0.4)</td>
<td>74.5 (+1.5)</td>
<td>37.7 (+0.2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Llama-2:13B</td>
<td>–</td>
<td>91.5</td>
<td>17.2</td>
<td>35.2</td>
<td>44.1</td>
<td>89.8</td>
<td>85.7</td>
</tr>
<tr>
<td>w: ReFACTOR</td>
<td>–</td>
<td></td>
<td>98.1</td>
<td>17.0 (-0.2)</td>
<td>34.7 (+0.5)</td>
<td>44.6 (+0.5)</td>
<td>90.0 (+0.2)</td>
<td>82.3 (-3.4)</td>
</tr>
<tr>
<td></td>
<td>Llama-2:7B</td>
<td>–</td>
<td>94.2</td>
<td>14.1 (+0.4)</td>
<td>31.8 (+0.3)</td>
<td>40.7 (+0.2)</td>
<td>89.8 (+0.0)</td>
<td>72.2 (+3.5)</td>
</tr>
<tr>
<td></td>
<td>Mistral-7B</td>
<td>Llama-2</td>
<td>94.2</td>
<td>14.1 (+0.4)</td>
<td>31.8 (+0.3)</td>
<td>40.7 (+0.2)</td>
<td>89.8 (+0.0)</td>
<td>72.2 (+3.5)</td>
</tr>
<tr>
<td></td>
<td>Llama-2:13B</td>
<td>–</td>
<td>105.0</td>
<td>13.6</td>
<td>32.5</td>
<td>42.5</td>
<td>89.1</td>
<td>75.3</td>
</tr>
<tr>
<td>w: ReFACTOR</td>
<td>Llama-2</td>
<td>86.5 (+0.9)</td>
<td>31.5 (-0.3)</td>
<td>38.3</td>
<td>88.1</td>
<td>81.3</td>
<td>39.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Llama-2:7B</td>
<td>–</td>
<td>75.0</td>
<td>19.3</td>
<td>37.0</td>
<td>43.8</td>
<td>90.1</td>
<td>78.4</td>
</tr>
<tr>
<td>w: ReFACTOR</td>
<td>–</td>
<td></td>
<td>80.0</td>
<td>20.2</td>
<td>40.0</td>
<td>49.7</td>
<td>90.9</td>
<td>91.7</td>
</tr>
<tr>
<td></td>
<td>Llama-2:13B</td>
<td>–</td>
<td>82.7</td>
<td>18.2</td>
<td>37.5</td>
<td>46.2</td>
<td>90.2</td>
<td>86.0</td>
</tr>
<tr>
<td>w: ReFACTOR</td>
<td>–</td>
<td></td>
<td>92.0</td>
<td>18.1 (-0.1)</td>
<td>37.0 (-0.3)</td>
<td>46.2 (+0.0)</td>
<td>90.3 (+0.1)</td>
<td>86.7 (+0.2)</td>
</tr>
<tr>
<td></td>
<td>Llama-2:7B</td>
<td>–</td>
<td>85.2</td>
<td>20.3 (+0.1)</td>
<td>39.8 (-0.2)</td>
<td>50.0 (+0.3)</td>
<td>91.2 (+0.3)</td>
<td>92.2 (+0.5)</td>
</tr>
<tr>
<td>w: ReFACTOR</td>
<td>–</td>
<td></td>
<td>92.1</td>
<td>19.0</td>
<td>39.9</td>
<td>51.2</td>
<td>91.0</td>
<td>94.3</td>
</tr>
<tr>
<td></td>
<td>Llama-2:13B</td>
<td>–</td>
<td>94.7</td>
<td>19.5</td>
<td>40.5</td>
<td>51.0</td>
<td>91.1</td>
<td>93.3</td>
</tr>
<tr>
<td>w: ReFACTOR</td>
<td>–</td>
<td></td>
<td>90.1</td>
<td>19.5</td>
<td>40.5</td>
<td>51.1</td>
<td>91.1</td>
<td>93.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># Examples</th>
<th>Error Types</th>
<th>Representative Question</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 / 200</td>
<td>Difficulty in parsing cell values via rule-based methods</td>
<td>The relevant numeric- or time-type columns are hard to parse (e.g., multiple numbers and text within one cell), thus ReFACTOR fail to generate related facts.</td>
<td></td>
</tr>
<tr>
<td>17 / 200</td>
<td>Complex user query causes difficulty in ranking related facts</td>
<td>Analyze the correlation between the size of the geographical area of a Gmina type and its population?</td>
<td>ReFACTOR employs the QA encoding model for fact ranking. However, it struggles to understand complex information needs from users, such as the “correlation between A and B”, and might consequently rank irrelevant facts higher.</td>
</tr>
<tr>
<td>13 / 200</td>
<td>Unsupported reasoning operations</td>
<td>Who are the top three coaches with the highest win percentages? Analyze their performance in the 2019-2020 season.</td>
<td>The table only contains “wins” and “overall games” columns. Models must compute the winning percentages independently. However, ReFACTOR does not support such rate calculations.</td>
</tr>
<tr>
<td>5 / 200</td>
<td>Other errors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>141 / 200</td>
<td>Successful cases</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 6: Human evaluation results (Likert Scale Scoring) of selected baselines on the test set. Five experts are enrolled to evaluate 50 predictions for each model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Faithfulness</th>
<th>Comprehension</th>
<th>Fluency</th>
</tr>
</thead>
<tbody>
<tr>
<td>BART</td>
<td>3.26</td>
<td>3.67</td>
<td>4.56</td>
</tr>
<tr>
<td>w. ReFACTOR 3.37 (+0.11)</td>
<td>3.72 (+0.05)</td>
<td>4.59 (+0.03)</td>
<td></td>
</tr>
<tr>
<td>OmniTab</td>
<td>3.30</td>
<td>3.58</td>
<td>4.52</td>
</tr>
<tr>
<td>w. ReFACTOR 3.45 (+0.15)</td>
<td>3.69 (+0.11)</td>
<td>4.52 (+0.0)</td>
<td></td>
</tr>
<tr>
<td>1-shot Mistral-7B 2.98</td>
<td>3.77</td>
<td>4.65</td>
<td></td>
</tr>
<tr>
<td>w. ReFACTOR 3.12 (+0.14)</td>
<td>3.82 (+0.05)</td>
<td>4.52 (-0.13)</td>
<td></td>
</tr>
<tr>
<td>1-shot Llama-2-70B 3.08</td>
<td>3.82</td>
<td>4.69</td>
<td></td>
</tr>
<tr>
<td>w. ReFACTOR 3.36 (+0.28)</td>
<td>3.99 (+0.17)</td>
<td>4.66 (-0.03)</td>
<td></td>
</tr>
<tr>
<td>0-shot GPT–5 3.65</td>
<td>3.94</td>
<td>4.66</td>
<td></td>
</tr>
<tr>
<td>w. ReFACTOR 3.84 (+0.19)</td>
<td>4.03 (+0.09)</td>
<td>4.74 (+0.08)</td>
<td></td>
</tr>
<tr>
<td>0-shot GPT–4 3.92</td>
<td>4.12</td>
<td>4.84</td>
<td></td>
</tr>
<tr>
<td>w. ReFACTOR 4.08 (+0.16)</td>
<td>4.15 (+0.03)</td>
<td>4.70 (-0.14)</td>
<td></td>
</tr>
<tr>
<td>1-shot GPT–3.5 3.84</td>
<td>4.20</td>
<td>4.86</td>
<td></td>
</tr>
<tr>
<td>w. ReFACTOR 3.95 (+0.11)</td>
<td>4.27 (+0.07)</td>
<td>4.84 (-0.02)</td>
<td></td>
</tr>
<tr>
<td>1-shot GPT–4 4.11</td>
<td>4.32</td>
<td>4.88</td>
<td></td>
</tr>
<tr>
<td>w. ReFACTOR 4.08 (-0.03)</td>
<td>4.35 (+0.03)</td>
<td>4.76 (-0.12)</td>
<td></td>
</tr>
</tbody>
</table>

We identify four common mistakes that current text generation models are likely to make (i.e., hallucination, factual incorrectness, user intent misunderstanding, and repetition), providing detailed examples and explanations for each type of common mistake in Table 8 in the Appendix.

5.5 ReFACTOR Analysis

We also undertake a human evaluation to examine the efficacy of ReFACTOR in generating query-relevant facts from tabular data. Specifically, we randomly sample 200 examples from QTSUMM validation set, and ask two human evaluators to evaluate each fact generated by ReFACTOR, determining its relevance to the query. 56.4% generated facts (528 out of 937) are labeled as “relevant”, suggesting an adequate coverage of ReFACTOR. To delve deeper into this, we also conduct a case study examining the failure cases, specifically those examples where less than two facts were annotated as “relevant”. We identified three kinds of common failure cases: (1) difficulty in parsing cell values via rule-based methods, (2) complex user query causes difficulty in ranking related facts, and (3) unsupported reasoning operations. We provide detailed examples and explanations in Table 5.

6 Conclusion

This paper defines a new query-focused table summarization task, and constructs a large-scale benchmark, QTSUMM. We investigate a set of strong baselines, including text generation, table-to-text generation, and large language models. Experimental results and manual analysis reveal that the new task presents significant challenges in table-to-text generation. Moreover, we propose a novel approach named ReFACTOR, to retrieve and reason over query-relevant information from tables, improving the faithfulness of generated summary.

Acknowledgements

We would like to dedicate this paper to the memory of Dr. Dragomir Radev. Dr. Radev’s leadership, guidance, and expertise were instrumental in shaping the direction and quality of this project. We appreciate the efforts of all annotators in constructing QTSUMM and conducting human evaluation. We are grateful to the Google TRC program for their support. We would also like to thank the anonymous reviewers and action editors for constructive discussions and feedback.
Limitations and Future Work

The baseline systems provided have a restricted maximum number of tokens they can accommodate (e.g., 1024 for all examined fine-tuned models), which prevents them from generating summaries for large tables that, when converted into a sequence, exceed the maximum number of tokens. To handle large tables (e.g., with more than 300 table cells), future work can apply neural models (Herzig et al., 2020; Liu et al., 2022b) to first filter out those query-irrelevant rows or columns.

Moreover, this paper demonstrates the effectiveness of using intermediate results obtained from explicit reasoning operations to mitigate the implicit reasoning issues. However, the proposed ReFACTO utilizes template-based method to generate facts. Although such template-based approach can ensure the factual correctness of generated facts, as discussed in Section 5.5, it might not cover all crucial facts for some complex user query. We believe following directions warrant further exploration: (1) Complex query decomposition. Our case study reveals that the TAPEX-based fact ranking module struggles with comprehending complex questions. To address this, future research could investigate LLM chain-of-thought methods to break down complex questions into more understandable and actionable sub-questions. (2) Tool usage. The predefined and template-based execution modules in the ReFACTO fact generation phase have their limitations. Recent studies (Schick et al., 2023; Lu et al., 2023; Paranjape et al., 2023; Gou et al., 2023; Qiao et al., 2023) highlight the impressive abilities of LLMs in making and utilizing tools for problem-solving. It would be intriguing to explore if LLMs can produce executable programs from scratch to derive query-relevant insights. (3) Explainable automated evaluation. In Section 5.3, a discrepancy between automated and human evaluation results is observed. Such discrepancies are concerning, as developers might opt for suboptimal systems for real-world applications if they solely rely on automatic metrics for comparing and ranking different text generation systems. Therefore, a more reliable and explainable automated evaluation system is required (Zhang and Bansal, 2021; Liu et al., 2023a,b; Jiang et al., 2023b).

Ethical Consideration

The source tables in QTSUMM were collected from LogicNLG (Chen et al., 2020a) and ToTTO (Parikh et al., 2020) datasets, which are publicly available under the MIT license and CC BY-SA 3.0 license, respectively. They both permit us to compose, modify, publish, and distribute additional annotations upon the original dataset.

For the external annotation of QTSUMM, we hired 17 graduate students majoring in STEM majors. We regard 1) creating three queries for one table, and validating the corresponding summaries annotated by others, and 2) composing a query-focused summary response as a unit task. And we paid around $1.5 for each unit task. For creative annotation rewards, we paid additional $0.5 for a query, and $1.5 for a summary. Averagely, an annotator can finish 7 unit tasks per hour after training and practicing. And the hourly rates are in the range of $9 and $13 based on the different working speed (above the local average wage of similar jobs). We recommended that annotators complete a maximum of 30 unit tasks per day in order to reduce pressure and maintain a comfortable pace. In total, the approximate working hours to annotate QTSUMM dataset was 1,400 hours. The whole annotation work lasted about 40 days.

References


Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. Critic: Large language models can self-correct with tool-interactive critiquing.


Dongfu Jiang, Yishan Li, Ge Zhang, Wenhao Huang, Bill Yuchen Lin, and Wenhui Chen. 2023b. Tigr-score: Towards building explainable metric for all text generation tasks.


Yixin Liu, Alexander R. Fabbri, Pengfei Liu, Yilun Zhao, Linyong Nan, Ruilin Han, Simeng Han, Shafiq Joty, Chien-Sheng Wu, Caiming Xiong, and Dragomir Radev. 2023a. Revisiting the gold standard: Grounding summarization evaluation with robust human evaluation.


Implementation Details

Input Data Serialization

The input contains a user query, and corresponding table data. For text generation and large language models (Section 5.1.1 & 5.1.3), we followed recent works on table-to-text generation (Liu et al., 2022b; Xie et al., 2022; Zhao et al., 2023c,a) to flatten the table data as $T = [\text{HEADER}]: h, [\text{ROW}]_1: r_1, \ldots, [\text{ROW}]_n: r_n$, where $h$ is table header, $r_i$ is the $i$-th table row. For text generation models, [HEADER] and [ROW] are special tokens indicating the region of table headers and rows respectively; while for LLMs, we set them as empty strings. We also separated headers or cells in different columns using a vertical bar |. In this way, the flattened table input can be fed directly into text generation models. For table-to-text generation models (Section 5.1.2), we followed their original data processing methods to input the query and table data.


A Implementation Details

Input Data Serialization

The input contains a user query, and corresponding table data. For text generation and large language models (Section 5.1.1 & 5.1.3), we followed recent works on table-to-text generation (Liu et al., 2022b; Xie et al., 2022; Zhao et al., 2023c,a) to flatten the table data as $T = [\text{HEADER}]: h, [\text{ROW}]_1: r_1, \ldots, [\text{ROW}]_n: r_n$, where $h$ is table header, $r_i$ is the $i$-th table row. For text generation models, [HEADER] and [ROW] are special tokens indicating the region of table headers and rows respectively; while for LLMs, we set them as empty strings. We also separated headers or cells in different columns using a vertical bar |. In this way, the flattened table input can be fed directly into text generation models. For table-to-text generation models (Section 5.1.2), we followed their original data processing methods to input the query and table data.
Reasoning | Example of Fact Templates | Example of Fact
---|---|---
Conjunction | The col that have CONDITION are executed_results. | The Player Name that have Country is Canada are Corey Conners, Nick Taylor, Adam Svensson.
Counting | executed_results col:1 have col:2 CONDITION:2. | 2 Game have Attendance greater than 10,235.
Temporal or Numerical Order | The col:1 ordered by col:3 are executed_results. The col:1, with col:2 CONDITION:2, ordered by col:3 are executed_results. | The Company ordered by Sales are Apple, Nvidia, Google, (...abbreviate...)
Temporal or Numerical Comparison | The col:1 that col:2 CONDITION:2 are executed_results. | The institutions that Founded year is earlier than 1860 are Adrian College, Michigan State University.
Numerical Operation (Sum, Avg) | The OPERATOR of col:1 with col:2 CONDITION:2 is executed_results. | The sum of Earning with Point is greater than 140 is 430,027.
Numerical Operation (Diff) | The difference between val:1 and val:2 in col is executed_results. | The difference between China and Canada in Gold is 16.

Table 7: 6 reasoning operations, along with fact template and examples, defined for the fact generation process of REFACTOR. Variable names indicate permissible instantiations. col denotes a column name, val denotes a cell value, and executed_results denotes the execution results of the function. OPERATOR is instantiated according to the specific reasoning operation, e.g., for “Numerical Operation”, OPERATOR is replaced with “sum” or “average”; CONDITION can be 1) a cell value from the i-th column, or 2) number/temporal comparison statement (e.g. “later than 1967”) if the i-th column is of number or date type.

Table Title: Top 10 Company in 2012

<table>
<thead>
<tr>
<th>Rank</th>
<th>Company Name</th>
<th>Industry</th>
<th>Revenue ($ Million)</th>
<th>Profit ($ Million)</th>
<th>Employees</th>
<th>Headquarters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Walmart</td>
<td>Retail</td>
<td>$559,151</td>
<td>$13,610</td>
<td>2,300,000</td>
<td>United States</td>
</tr>
<tr>
<td>2</td>
<td>State Grid</td>
<td>Electricity</td>
<td>$386,618</td>
<td>$5,580</td>
<td>896,360</td>
<td>United States</td>
</tr>
<tr>
<td>3</td>
<td>Amazon</td>
<td>Retail</td>
<td>$386,064</td>
<td>$21,331</td>
<td>1,298,000</td>
<td>United States</td>
</tr>
<tr>
<td>4</td>
<td>National Petroleum</td>
<td>Oil and gas</td>
<td>$283,958</td>
<td>$4,575</td>
<td>1,242,245</td>
<td>China</td>
</tr>
<tr>
<td>5</td>
<td>Sinopec Group</td>
<td>Oil and gas</td>
<td>$283,726</td>
<td>$6,205</td>
<td>553,833</td>
<td>China</td>
</tr>
<tr>
<td>6</td>
<td>Apple</td>
<td>Electronics</td>
<td>$274,515</td>
<td>$57,511</td>
<td>147,000</td>
<td>United States</td>
</tr>
<tr>
<td>7</td>
<td>CVS Health</td>
<td>Healthcare</td>
<td>$268,706</td>
<td>$7,179</td>
<td>296,500</td>
<td>United States</td>
</tr>
<tr>
<td>8</td>
<td>UnitedHealth</td>
<td>Healthcare</td>
<td>$257,141</td>
<td>$15,403</td>
<td>330,000</td>
<td>United States</td>
</tr>
<tr>
<td>9</td>
<td>Toyota</td>
<td>Automotive</td>
<td>$256,722</td>
<td>$21,190</td>
<td>366,263</td>
<td>Japan</td>
</tr>
<tr>
<td>10</td>
<td>Volkswagen</td>
<td>Automotive</td>
<td>$253,965</td>
<td>$10,104</td>
<td>662,575</td>
<td>Germany</td>
</tr>
</tbody>
</table>

Using the information from the table, summarize a paragraph-long response to the following user query: Which company earns the highest profit in the Oil and Gas industry, and how does it compare to the most profitable company overall?

Number of Relevant facts of: 1. The Company Name ordered by Profit ($ Million) is Apple, Amazon, Toyota, UnitedHealth, Walmart, CVS Health, Sinopec Group, State Grid, National Petroleum. 2. The Company Name, with Industry is Oil and gas, ordered by Profit ($ Million) is Sinopec Group, National Petroleum. 3. The sum of Profit with Industry is Oil and gas is 10780. 4. The difference between Apple and Sinopec Group in Profit is 51306. 5. The average of Profit with Industry is Oil and gas is 5390.

Using the information from the table and relevant facts of table, summarize a paragraph-long response to the following user query: Which company earns the highest profit in the Oil and Gas industry, and how does it compare to the most profitable company overall?

Figure 4: An example of LLM zero-shot prompt prefix REFACTOR for the QTSUMM task.

Figure 5: An example of LLM zero-shot prompt prefix REFACTOR for the QTSUMM task.
Q: Which driver performed better, Tracy or Haberfeld, and why?

**System output:**
Paul Tracy performed better than Mario Haberfeld, as evidenced in their results from the 2004 Centrix Financial Grand Prix of Denver. In the race, Tracy finished in second place, while Haberfeld placed in eighth. Additionally, Tracy belongs to Forsythe Racing, which has had a more successful track record compared to Haberfeld’s Walker Racing. Additionally, Tracy was able to maintain his position in the race for the full 90 laps, whereas Haberfeld was only able to make it to 86 laps. Based on these results, it is clear that Tracy was more competitive compared to Mario Haberfeld.

**Reference:**
Paul Tracy is more competitive. Tracy finished the race in second place with a time of +7.446 seconds, whereas Haberfeld finished in eighth place with a time of +24.141 seconds. This demonstrates that Tracy was able to navigate the track much faster than Haberfeld and place higher in the race. Additionally, Tracy earned 28 points while Haberfeld earned only 15 points, which confirms that Tracy is a stronger competitor. In conclusion, Tracy is the more competitive driver.

**Error analysis:**
- **Error Type:** Hallucination
- **Explanation:** Mention facts that are not shown in the table.
- **Analysis:** There is no information indicating that Forsythe Racing has a more successful track record compared to Walker Racing, and the model incorrectly states that Haberfeld completed only 86 laps instead of the correct 90 laps.

---

Q: According to the voting result, which representative election in North Carolina districts was the most competitive, and why?

**System output:**
The most competitive representative election in North Carolina districts was the one for the 7th district, in which incumbent Larry Kissell (D) lost re-election to Richard Hudson (R). This was the only race which was decided by less than 10 percentage points, with Hudson winning 54.1% of the vote to Kissell’s 45.9%. The close margin of the race demonstrates that both candidates were competitive, and it was an intense contest until the very end.

**Reference:**
The race in the North Carolina 7th district was the most competitive, as the Democratic incumbent Mike McIntyre won by a slim margin, with only a 0.2% difference between him and his Republican challenger David Rouzer. Furthermore, this election was the only one among all North Carolina districts in 2012 that resulted in a margin of victory within less than 1%.

**Error analysis:**
- **Error Type:** Factual Incorrectness
- **Explanation:** Present information that contradicts the source table.
- **Analysis:** The most competitive representative election should be the North Carolina 7th district, rather than the 8th district, in which the difference between the two rivals was only 0.2%

---

Q: Analyse the countries in East Asia with Human Development Index (HDI) higher than 0.8.

**System output:**
The countries in East Asia with HDI higher than 0.8 are Hong Kong (PRC), Japan, and South Korea. Hong Kong has the highest HDI at 0.898, followed by Japan at 0.901. Macau has the lowest HDI at 0.653, and North Korea has the highest at 0.897. All of these countries have populations greater than 1.8 million, with the highest population density being in Japan at 337 people per km square.

**Reference:**
The countries in East Asia with an HDI higher than 0.8 are Hong Kong (PRC), Japan, and South Korea. Hong Kong has an HDI of 0.898 and a population density per square kilometer of 6390. Japan has an HDI of 0.901 and a population density per square kilometer of 337. South Korea has an HDI of 0.897 and a population density per square kilometer of 500. All three countries have a comparatively high population density which is likely to be a factor in their high HDI.

**Error analysis:**
- **Error Type:** User Intent Misunderstanding
- **Explanation:** Include information that is irrelevant to the user question.
- **Analysis:** The query does not ask for country with lowest HDI, or any country with HDI lower than 0.8. The system output also displays many other errors such as hallucination (confusing South Korea which is Korea) and factual incorrectness (Japan has highest HDI).

---

Q: What are the buildings that have been the tallest in Portland, Oregon between 1927 and 1965? Summarize the basic information of these buildings.

**System output:**
The buildings that have been the tallest in Portland, Oregon between 1927 and 1965 are the Public Service Building, Hilton Portland Hotel, and Harrison West Condominium Tower. The Public Service Building was located at 920 Sw Sixth Avenue and has been the tallest building in Portland, Oregon. It stands at a height of 220 feet (67 meters) and has 16 floors. The Hilton Portland Hotel is located at 921 Sw Sixth Avenue and has been the tallest building in Portland, Oregon.

**Reference:**
Between 1927 and 1965, the tallest buildings in Portland, Oregon were the Public Service Building, Hilton Portland Hotel, and Harrison West Condominium Tower. Specifically, Public Service Building was built in 1927, was 220 feet / 67 meters tall, had 16 floors, and held the title of tallest building until 1962. Hilton Portland Hotel was built in 1963, was 241 feet / 73 meters tall, had 22 floors, and held the title of tallest building until 1965. Harrison West Condominium Tower was built in 1965, was 256 feet / 78 meters tall, had 25 floors, and held the title of tallest building until 1969.

**Error analysis:**
- **Error Type:** Repetition
- **Explanation:** Generate repetitive information.
- **Analysis:** The information of these buildings being the tallest in Portland, Oregon has been mentioned repetitively throughout the system output, while the system fail to also distinguish them (until which year each of them was the tallest respectively).

---

Table 8: Case study for common errors made by Flan-T5-large w/o. ReFACTOR. The colored text highlights problematic parts of the system output.

1172