MultiTabQA: Generating Tabular Answers for Multi-Table Question Answering

Vaishali Pal[1,2] **Andrew Yates**[1] **Evangelos Kanoulas**[1] **Maarten de Rijke**[1]
[1]University of Amsterdam, The Netherlands
[2]Discovery Lab, Elsevier, The Netherlands
v.pal, a.c.yates, e.kanoulas, m.derijke@uva.nl

Abstract

Recent advances in tabular question answering (QA) with large language models are constrained in their coverage and only answer questions over a single table. However, real-world queries are complex in nature, often over multiple tables in a relational database or web page. Single table questions do not involve common table operations such as set operations, Cartesian products (joins), or nested queries. Furthermore, multi-table operations often result in a tabular output, which necessitates table generation capabilities of tabular QA models. To fill this gap, we propose a new task of answering questions over multiple tables. Our model, MultiTabQA, not only answers questions over multiple tables, but also generalizes to generate tabular answers. To enable effective training, we build a pre-training dataset comprising of 132,645 SQL queries and tabular answers. Further, we evaluate the generated tables by introducing table-specific metrics of varying strictness assessing various levels of granularity of the table structure. MultiTabQA outperforms state-of-the-art single table QA models adapted to a multi-table QA setting by finetuning on three datasets: Spider, Atis and GeoQuery.

1 Introduction

Question answering (QA) over multiple tables aims to provide exact answers to natural language questions with evidence from one or more tables (Jin et al., 2022). This is in contrast to single-table QA, which has been the focus of tabular QA research to date (Liu et al., 2021; Nan et al., 2021; Zhu et al., 2021; Herzig et al., 2020). Even though groups of related tables are ubiquitous in real-world corpora, such as relational databases or tables in a web page, multi-table QA remains a largely unexplored area. To address this gap, we propose a new task of answering questions over multiple tables. Our multi-table QA model, MultiTabQA,[1] addresses

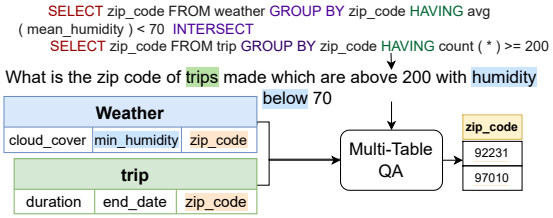


Figure 1: Multi-table QA. The QA model generates a tabular answer from either a natural language question or an SQL query and one or more tables as input context.

novel challenges introduced by multi-table context. These include complex queries involving chains of reasoning, disambiguation of relevant table names at each reasoning step, and generating a final table as answer. It also leads to novel question-types that are unnatural in a single-table setting. For instance, questions involving operations specific to multiple tables, such as Cartesian products (*outer joins*, *inner joins*) and set operations (such as *intersect*, *union*, *in*), are unique to and common in a multi-table scenario. Furthermore, such multi-table operations often result in a tabular answer and they necessitate table generation capabilities of the QA model.

Figure 1 depicts an example of a question involving two tables, *I would like to know the zip code of trips taken above 200 with humidity below 70*, and its associated input tables, *Weather* and *trip*. A multi-table QA model is expected to disambiguate records from different tables (the question phrase *zip code of trips* grounds the column *zip_code* of Table *trip*; the question phrase *humidity below 70* grounds column *min_humidity* of Table *Weather*), learn associations among inter-table columns (*zip_code* in both tables) and intra-table columns (*min_humidity* and *zip_code* in the *Weather* table), and finally compute the required operations (*intersect*, *count*) and generate the tabular answer.

Recent work on tabular QA can be categorized into two major directions: (i) Semantic parsing-based techniques (Pasupat and Liang, 2015;

[1]Code and data are at: https://github.com/kolk/MultiTabQA

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 6322–6334
July 9-14, 2023

Zhong et al., 2017; Cai et al., 2022), which have been the dominant approach to answering multi--table complex questions. Such methods transform a natural question to a logical form, which is used to query a relational database to extract the answer. However, these techniques are restricted to relational databases and cannot be applied to tables from other sources such over web tables, tables in text documents, and any non-normalized tables. Additionally, they require expensive and expert human annotations (Yu et al., 2018; Lee et al., 2021) formulating SQL queries from natural questions. (ii) Modeling the problem as a sequence generation/classification task (Yin et al., 2020; Zhang et al., 2020; Herzig et al., 2020; Zhu et al., 2021; Liu et al., 2021; Cheng et al., 2021b; Nan et al., 2021; Ma et al., 2022; Pal et al., 2022; Jin et al., 2022), where an end-to-end trained neural model is not only responsible for question/query understanding but also table reasoning. Existing end–to-end neural models are either classification-based (Herzig et al., 2020; Zhu et al., 2021), where the model detects the answer span and classifies one table operator associated with the span, or they are sequence generation-based (Nan et al., 2021; Zhang et al., 2020; Liu et al., 2021), where the model generates the answer as a span of text in an auto-regressive manner.

Our work focuses on the latter direction of research. We train a neural model to mimic a semantic parser and generate the answer. A clear distinction of our work compared to existing end-to-end models is that our proposed model, MultiTabQA, does not operate in the constrained setting of a single input table, but can accommodate one or more tables in the input and the associated multi-table operators. Additionally, MultiTabQA performs the task of structured table generation, which imposes structure aspects to the generated output such as table schemas, alignments of rows and columns, relationships between column-headers and column values. Generating structured tables as output requires table-specific evaluation metrics which we define and use to evaluate the generated tables. To effectively train the model, we generate a pre-training dataset with multi-table SQL queries and tabular answers built over an existing semantic parsing dataset, Spider (Yu et al., 2018). Our dataset consists of $132,645$ samples comprising of SQL queries, associated natural language questions, input tables, and tabular answers. To the best of our knowledge, this is the first work to address the task of multi-table QA and generate tabular output.

Our main contributions can be summarized as:

(1) We fill-in the gap of existing tabular QA methods, which operate only on single tables, by proposing a new task of answering questions over multiple tables. Our work increases the breadth of question types that can be handled by single tabular QA methods.
(2) Our proposed multi-table QA model generates structured tables imposed by multi-table operations. Table generation introduces generation challenges such as maintaining row-column alignment, table-header generation, etc.
(3) We release a multi-table pre-training dataset comprising of $132,645$ samples of SQL queries and tabular answers.
(4) We introduce table generation metrics that capture different levels of granularity and strictness to evaluate our proposed model.

2 Methodology

We frame multi-table question answering as a sequence-to-sequence task and train an auto-regressive transformer encoder-decoder model to generate the tabular answer. Given a question Q consisting of a sequence of k tokens $q_1, q_2, \ldots, q_k$ and a set of N tables, $T_N = \{t_1, t_2, \ldots, t_n\}$, the goal of the multi-table QA model is to perform chains of *operations* over T_N, constrained by Q, and generate a table T_{out}. The model always generates a table, T_{out}, which can be single celled for scalar answers, single rowed or columned for list-based answers, and multiple rows and columns for tabular answers. In all cases, the model also generates column headers revealing important semantics associated with the generated values.

Training approach. We follow a curriculum learning approach (Bengio et al., 2009) by sequentially increasing the complexity of tasks to train MultiTabQA. The first stage of training is a pre-training step where the training objective is two-fold: (i) learn to generate correct tabular answers from SQL, and (ii) understand the associations between related input tables. The final training stage is fine-tuning where the model learns to understand natural language questions with their inherent ambiguity in addition to retaining its ability of reasoning over tables and generating a tabular answer. We discuss the training process in detail in Section 4.

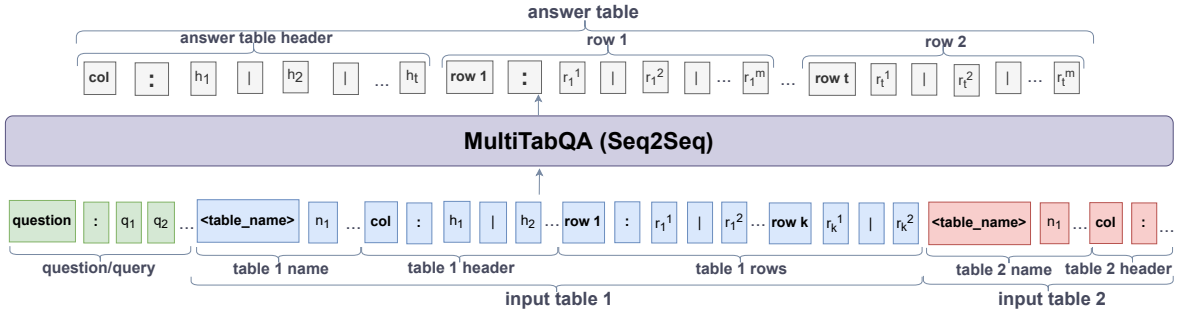


Figure 2: Architecture of MultiTabQA model. Given a natural language question/SQL query and the associated tables as an input sequence, the seq2seq model performs tabular reasoning and generates a tabular answer. Start of an input table is identified with keyword **<table_name>** which also indicates that the next tokens comprises the table name. **col:** indicates that the next tokens are table headers. Rows in a table are identified with keyword **row i:**, columns are separated by |.

Model input/output. The input to the model is a sequence comprised of the query or the natural language question, followed by a sequence of input tables, represented by the table name and the corresponding flattened table. Table names are important for disambiguating tables in multi-table QA setting. Specifically, the input sequence is represented as $question\ [table_1\ rep]\ [table_2\ rep] \ldots [table_n\ rep]$ where $[table_i\ rep]$ is the representation of the i-th table. As depicted in Figure 2, the i-th table is flattened in row-major format and represented as

$$\textbf{<table\_name>:}\ n_1\ n_2\ \mid \textbf{col:}\ h_1 \mid h_2 \mid \ldots \mid h_k$$
$$\textbf{row 1:}\ r_1^1 \mid \ldots \mid r_1^m \ldots \textbf{row k:}\ r_k^1 \mid \ldots \mid r_k^m,$$

where $n_1, \ldots, n_2$ is the sequence of table name tokens, h_j is j-th column header, r_m^i is the i-th row and m-th column cell. The boldface words are keywords specifying semantics of the next tokens. The output of the model is also a flattened table in row-major format, i.e., $[table_{ans}\ rep]$, but without a table name. As depicted in Figure 2, the generated table, $[table_{ans}\ rep]$, is of the form:

$$\textbf{col:}\ h_1 \mid h_2 \mid \ldots \mid h_k\ \textbf{row 1:}\ r_1^1 \mid \ldots \mid r_1^m$$
$$\textbf{row 2:}\ r_2^1 \mid \ldots \mid r_2^m \ldots \textbf{row k:}\ r_k^1 \mid \ldots \mid r_k^m.$$

3 Dataset

To effectively train a multi-table QA model, the dataset needs to cover three aspects: (i) multi-table context, (ii) tabular answers, and (iii) natural questions. Given the absence of large-scale datasets covering all three aspects, we transform existing semantic parsing and single-table QA datasets to focus on a single aspect before training with samples covering all three aspects.

3.1 Single table pre-training dataset

One of the sub-tasks of pre-training is to generate tabular answers. We hypothesize that tuning the model to generate tables may lead to a warm-start and better convergence in a multi-table QA setting. To enable such experiments, we modify the large-scale single-table QA Tapex pre-training dataset (Liu et al., 2021) to accommodate tabular answers. The dataset contains $1,834,419$ samples of query, input table and factoid answers. The tables in the dataset are not named as there is no need for table disambiguation in a single table setting. The SQL queries are semi-formal (do not contain the `FROM` clause with a table name) and cannot be used to query a real SQL database. We insert a placeholder table name in the queries and the corresponding input tables to extract the tabular answer from the database. Transforming the factoid answers to tables leads to single-celled or single-rowed tables. The modified dataset helps the model to understand simple tables and reason over semi-formal queries to generate simple tables.

3.2 Multi-table pre-training dataset

We develop a multi-table pre-training dataset over the database of Spider (Yu et al., 2018). Spider is a cross-domain complex semantic parsing dataset for text-to-SQL translation. It consists of $10,181$ questions and $5,693$ SQL queries. The questions are over 200 databases of multiple tables covering 138 different domains. The training, development and test splits do not contain overlapping databases to test a model's generalizability to new databases.

We first adapt the existing samples of Spider for our task. We use the ground-truth SQL queries of Spider as input query for pre-training over multiple tables. We automatically extract all input table

names from the SQL query and retrieve the input tables[2] from the relational database. The query, extracted table names, and retrieved tables are inputs to our multi-table QA model. We extract the answer table with the SQL query by querying the relational database. Answer table headers reveal important semantics of the associated column values such as the numeric operation (*average*, *sum*, etc.), numeric scales (million, thousand, kms, meters, etc.), or entity facets (name, date, etc.). This process generates 3816 samples comprising of *query*, *question*, *table_names*, *tables* and *answer*.

We further augment the modified Spider dataset with 132,645 samples of synthetic queries. This leads to an augmented multi-table pre-training dataset of 136,461 unique training samples comprising of 3816 Spider samples and 132,645 synthetic samples. The validation set comprises of 536 samples from the Spider validation set preprocessed as described above to adapt to our task.

Existing work on semantic parsing (Shi et al., 2020; Yu et al., 2021) have utilized hand-crafted templates to generate large-scale corpora of synthetic queries, but are constrained in their coverage with no multi-table operations (Shi et al., 2020) or limited coverage with no table *joins* and lacking diversity in *set* operations (Yu et al., 2021). This motivates us to generate our augmented pre-training dataset for multi-table QA using multi-table SQL templates.

Our synthetic queries are generated from 45 manually crafted templates over the Spider database and hand-crafted rules for operation types. The query templates have placeholders for aggregation, relational operations, table name and headers which are randomly assigned during query generation process. For example, to generate multi-table *join* queries, we instantiate the templates by randomly choosing tables from a database with at least one common header. For *set* operations, all tables participating in a multi-table query requires all table headers to match. We design SQL templates in increasing order of complexity starting with simple SQL templates and progressively adding components which increases its complexity. For example, for single-table queries, we use the simplest template "*SELECT * FROM {table_name}*" whereas for multi-table templates such as *joins*, the simplest template is "*SELECT T1.{table1_cols}, T2.{table2_cols} FROM {table_name1} as T1 JOIN {table_name2} as T2 ON T1.{common_col} = T2.{common_col}*". We progressively add SQL components such as aggregations, *where* conditions, *group by* and *having* clauses to generate templates of increasing complexity. This process results in 14 templates for *joins*, 4 templates for each set operation: *intersect*, *union* and *except*. To avoid catastrophic forgetting for single table queries, we also instantiate 14 single-table queries with increasing complexity.

Quality control. We ensure correctness of the synthetic samples by discarding SQL queries that executes to an error or empty table. We also apply the process on the modified Spider, Atis and GeoQuery data to discard SQL query and the corresponding natural language question to ensure that all questions are answerable.

3.3 Multi-table QA dataset

We fine-tune and evaluate our model on the natural language questions of semantic parsing datasets: Spider, GeoQuery (Zelle and Mooney, 1996), and Atis (Price, 1990; Dahl et al., 1994). GeoQuery is a semantic parsing dataset to query into a database of United States geography.[3] Atis is a semantic parsing dataset[4] with a collection of 4,379 questions, corresponding SQL queries and a relational database to a flight booking system (Iyer et al., 2017). Similar to the Spider dataset processing described in Section 3.2, we first extract the input table names from the available SQL queries and query the relational database for the input tables.[5] We also extract the tabular answers using the SQL queries. We discard any samples that executes to an error or empty table. We use the corresponding natural language question for each SQL query as the user utterance for fine-tuning. This results in 6,715 training samples and 985 validation samples for Spider. We also process the 600 GeoQuery samples provided in (Iyer et al., 2017) to create a subset of 530 training samples, 49 validation samples and 253 test samples. We process and generate an Atis subset of 384 training samples, 45 evaluation samples and 86 test samples. We discard Atis queries with very large input tables (with > 10,000 rows). This restriction enables us to correctly evaluate question answering capabilities of a model by

[2]We use SQLite3 and pandas for extracting tables.

[3]This data is made available under under GPL 2.0 license.

[4]This data is made available under MIT license.

[5]We preprocess the Atis and GeoQuery data samples available at https://github.com/sriniiyer/nl2sql/tree/master/data.

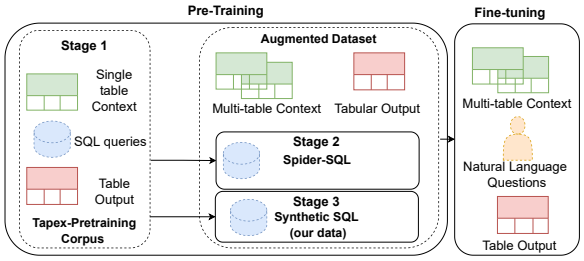


Figure 3: Four stage training procedure. The first three stages are pre-training, followed by fine-tuning.

ignoring samples with truncated input sequences including entire input tables from the second table onward. Truncation of tables leads to incorrect answers for any numeric operation such as *average*, *intersect* and the evaluation scores would no longer reflect reasoning capabilities of the model.

4 Training

We follow a curriculum learning approach by sequentially training the model on sub-tasks of increasing complexity as depicted in Figure 3. Broadly, we first pre-train the seq2seq model to mimic a SQL parser and further fine-tune it on the downstream multi-table QA task. Pre-training the model on unambiguous SQL queries leads to better convergence and warm-start for the closely related downstream multi-table QA task. We further segregate the pre-training by first addressing the simpler sub-task of generating tables from single table queries. This is immediately followed by pre-training on multi-table query answering where complex SQL queries are utilized to train the model to learn multi-table associations from unambiguous complex queries, reason over the tables and generate tabular answer. The final stage of training is the downstream multi-table QA from natural language questions. Natural language introduces ambiguity, ellipses and co-references which increases complexity and is thus the final stage of training. For each stage, we choose the model with the best table exact match accuracy on the corresponding validation set, defined in Section 5, as the initialization for training the next stage.

4.1 Pre-training

Pre-training of MultiTabQA is conducted in two stages in a curriculum learning fashion: Stage 1 is single table QA where the model learns to generate tabular answers from relatively simple SQL queries. Stage 2 is multi-table QA where the model trained in Stage 1 is further tuned for multi-table SQL QA.

Stage 1. We first train MultiTabQA on the task of generating tables from SQL queries over single tables. The tabular answer to be generated is simple and single-columned. For this stage, we use the modified Tapex pre-training corpus described in Section 3.1. We train the model on $1,834,419$ samples for two epochs. This stage provides a good initialization for multi-table QA in the next stages.

Stage 2 + Stage 3. We further pre-train the model on multi-table QA. For this, we tune our model on SQL queries from the modified Spider and synthetic dataset. We tune with only the modified Spider SQL samples *Stage 2*, and tuning with only the synthetic dataset *Stage 3*. We utilize the larger augmented dataset comprising of the modified Spider SQL (Stage 2) and our synthetic samples (Stage 3) as described in Section 3.2 to train the final pre-trained model for 30 epochs. We call this setting *Stage 2+3*. We compare these three multi-table pre-training settings in Section 6.

4.2 Fine-tuning

The final stage of training is fine-tuning the pre-trained model on natural language questions. Natural questions are ambiguous compared to formal SQL and used at the last stage of training. We fine-tune the pre-trained model on the $6,715$ natural questions, extracted input and output tables for Spider as described in Section 3 and evaluate on 985 samples of the validation set. To observe the performance of the pre-trained model on out-of-domain database tables, we also fine-tune the pre-trained model on Atis and GeoQuery datasets. For all the fine-tuning datasets, we train for 60 epochs.

5 Evaluation metrics

While denotation accuracy has been widely used in semantic parsing (Pasupat and Liang, 2015; Zhong et al., 2017; Cai et al., 2022), it is not directly applicable for our task where tabular input encoding, reasoning, and generation are performed by the same model. Evaluating the answer table not only requires matching the generated values but also the table structure. Moreover, tables store factual information such as named entities, dates, numbers, etc in an ordered manner. This makes lexical metrics measuring surface form overlap more suitable than semantic metrics measuring the underlying meaning of paraphrased sequences. Moreover, table components such as rows, columns and cells

Dataset	Model	Table EM (%)	Row EM (%)			Column EM (%)			Cell EM (%)		
			P	R	F1	P	R	F1	P	R	F1
Spider	tapex-base	18.99	17.28	19.83	18.27	19.75	19.39	19.57	23.15	27.71	25.03
	MultiTabQA	**25.19***	**22.88†**	**24.64***	**23.70***	**26.86***	**26.76***	**26.81***	**28.07†**	**31.23***	**29.55***
GeoQ	tapex-base	39.84	22.43	30.74	24.89	39.48	39.76	39.62	21.98	30.88	24.67
	MultiTabQA	**52.22***	**72.39***	**46.90***	**41.38***	**52.10***	**52.22***	**52.16***	**37.16†**	**46.92***	**41.33***
Atis	tapex-base	72.20	**57.07†**	57.69	**55.08**	**72.20†**	72.20	72.20	**57.07†**	57.69	**54.48**
	MultiTabQA	**73.88†**	38.29	**92.19***	54.36	69.55	**75.24†**	**72.29**	38.16	**92.56***	54.16

Table 1: Average scores of models fine-tuned on 5 different seeds with Multitable-Natural Questions (NQ) datasets. tapex-base is used as baseline while MultiTabQA is our fine-tuned model. Table EM indicates table exact match accuracy. For all other table units (row, column, and cell), P is Precision, R is Recall, and F1 is F1 score for exact match metric. An (*) denotes significance at $p < 0.005$ and an (†) denotes a significance at $p < 0.05$ for t-test.

are standalone units which capture different levels of semantics and relationships with the surrounding table component. For example, rows capture data records while columns capture the features of each record. Cells capture the lowest level of self-contained facts and requires complete match with the target. For example, a cell with the entity "United Kingdom" should not be partially matched with the predictions "United Nation", "United" or "Kingdom". Similarly, a numeric value such as "123.45" should not be partially matched with "12.45", "23.45" or "12". Numeracy pose a challenge to seq2seq models (Nogueira et al., 2021; Pal and Baral, 2021), especially in the extrapolation setting where semantic match of unseen numbers may not be an ideal. Considering all these factors, we focus on lexical match to measure model effectiveness.

Table exact match. We define *table exact match Accuracy (Table EM)* as the percentage of predicted tables which exactly matches the target tables. Table exact match evaluates ordering of rows, columns and table headers and exact lexical matching of table values. It is a strict binary measure which treats partial matches as incorrect. However, many queries do not impose ordering among columns or rows, and strict table exact match may not be the ideal indication of model efficacy. To measure partial correctness, we treat rows, columns and cells as units at varying levels of granularity which have ordered associations among the values within the unit. We evaluate partial correctness with exact match of rows, columns and cells.

Row exact match. To relax the strict criterion of table exact match, we first measure correctness on table rows. Row exact match does not consider ordering of rows in the generated table but requires ordering of values within the row. We define a correctly generated row to be a predicted row that exactly matches any target rows in the target table. *Row exact match precision* is the percentage of correctly generated rows among all the predicted rows in the evaluation dataset. *Row exact match recall* is the percentage of correctly generated rows among all the target rows in the evaluation dataset.

Column exact match. Unlike rows, which represent records in relational databases, columns represent attributes where column header provides semantic meaning to the values. Hence, a correct column is defined as a generated column that first exactly matches a target column header and further the column values. Column exact match measures ordering of values within a column. *Column exact match precision* is the percentage of correctly generated columns among all generated columns in the evaluation set. *Column exact match recall* is the percentage of correctly generated columns among all target columns in the evaluation set.

Cell exact match. *Cell exact match* is the most relaxed measure of model efficacy at the lowest level of granularity (cells) where table structure is not measured. A cell is correct if it matches any cell in the corresponding target table. *Cell exact match precision* is the percentage of correctly predicted cells among all predicted cells in the dataset. *Cell exact match recall* is the percentage of correctly predicted cells among all target cells in the dataset.

6 Experimental setup and results

We use tapex-base (Liu et al., 2021) as the base model for all our experiments. tapex-base is a single table question answering model (140M parameters) trained to approximate table reasoning by pre-training to mimic an SQL parser. For both the pre-training and fine-tuning process, we use a batch size of 8 and gradient accumulation of 32 to

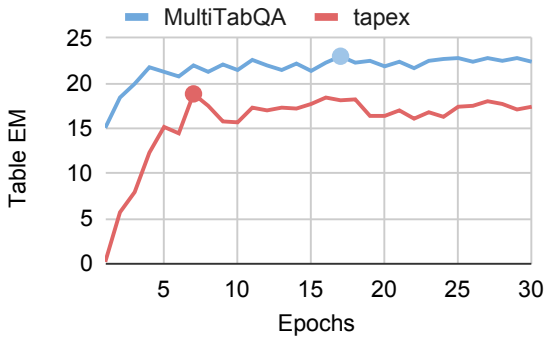


Figure 4: Validation table exact match scores of `MultiTabQA` vs. `tapex-base` on Spider evaluation set natural language questions during fine-tuning. The points are highest validation scores for each model.

emulate an effective batch size of 256, a learning rate is $1e^{-9}$. The maximum sequence length of both encoder and decoder is set to 1024. We run all our pre-training experiments on four A6000 48GB GPUs and fine-tuning on one A6000 GPU.

We observe from Figure 4 that the three stage pre-training leads to a warm-start for fine-tuning and better convergence compared to the baseline `tapex-base`. For our experiments, we compare the effectiveness of the MultiTabQA model with fine-tuned `tapex-base` on the 6,715 natural questions from Spider. The fine-tuned `tapex-base` acts as baseline for studying the adaptability of state-of-the-art single table model to a multi-table setting. We report the mean scores of 5 training runs initialized with different seeds in Table 1. We conduct statistical significance test (t-test) on the mean scores of the 5 runs and report the significance with $p < 0.05$ and $p < 0.005$. We observe that our multi-stage training process leads to improvement in scores on all table exact match accuracy across all datasets compared to fine-tuned `tapex-base`. The difference in table exact match is highest for GeoQuery where MultiTabQA outperforms `tapex-base` by 12.38%, Spider by 6.20% and Atis by 1.68%. For F1 and Recall scores on row, column and cell exact match, a similar pattern is observed where MultiTabQA outperforms `tapex-base` on all datasets. MultiTabQA outperforms `tapex-base` by 5.43% on row F1, 7.24% on column F1, and 4.52% on cell F1 for Spider. On GeoQuery, MultiTabQA outperforms by 16.49% on row F1, 12.54% on column F1 and 16.66% on cell F1 scores. All results on Spider and GeoQuery are significant with a p-value less than a critical value of 0.05 indicating strong evidence that MultiTabQA is a superior model. On Atis, we observe that MultiTabQA underperforms on precision but outperforms on recall by a large margin. The difference in recall is larger than precision indicating that MultiTabQA generates more target rows, columns and cells of Atis correctly (higher recall) and hallucinates spurious rows and cells (lower precision). However, the F1 scores are better for MultiTabQA. `tapex-base` is unable to correctly generate target rows, cells and columns (lower recall), but the few generated ones are correct (higher precision). The low number of test samples (85) of Atis and variations in the hallucinations in different runs makes the precision scores statistically non-significant. However, the recall scores provide very strong evidence ($p < 0.005$) of the superiority of MultiTabQA in generating correct table units compared to `tapex-base`.

Qualitative analysis. Multi-table QA models must perform numeric reasoning, understand multi-table schemas and comprehend natural language. A success case also depicts this. For the question *how many likes does kyle have?* with 2 input tables:

highschooler

id	name	grade
1510	jordan	9
...	...	...
1934	**kyle**	12
1661	logan	12

likes

student_id	like_id
1689	1709
...	...
1501	**1934**
1934	1501

with

target:

count(*)
1

and prediction:

count(*)
1

, MultiTabQA identifies inter-table association of column *id* of table *highschooler* and column *student_id* of table *likes*. It correctly disambiguates the lexical occurrence of 1934 in columns *like_id* and *student_id* and correctly performs *count*.

A failure case also illustrates the challenges: for the question *find the average weight for each pet type* with input table:

PetID	PetType	pet_age	weight
2001	cat	3	12.0
2002	dog	2	13.4
2003	dog	1	9.3

with

target:

avg(weight)	PetType
12.0	cat
11.35	dog

and

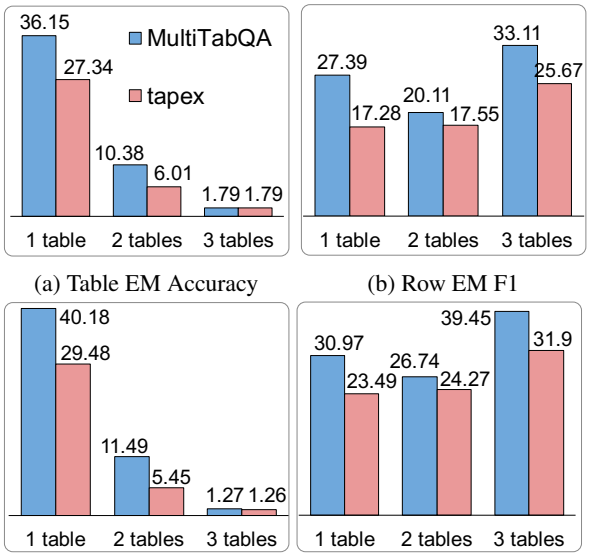
prediction:

PetType	avg(weight)
cat	12.0
dog	13.4

, MultiTabQA swaps the ordering of the 2 columns and fails to compute *average* leading to an incorrect measure by table exact match. The column, row and cell metrics (precision, recall and F1) measure

Pre-training stages	Query type	Table EM(%)	Row (%)			Column (%)			Cell (%)		
			P	R	F1	P	R	F1	P	R	F1
2	SQL	21.46	18.60	18.88	18.74	21.98	21.90	21.94	24.19	25.89	25.01
1+2		20.52	14.13	20.06	16.58	18.87	20.87	19.82	19.24	25.83	22.05
1+2+3		**29.10**	**23.15**	**25.62**	**24.32**	**31.66**	**31.50**	**31.58**	**29.95**	**32.92**	**31.36**
2	NL	19.41	16.51	19.48	17.87	20.13	20.11	20.12	21.12	26.55	23.52
1+2		20.12	11.67	21.09	15.03	19.54	19.97	19.76	16.26	29.22	20.90
1+2+3		**24.49**	**24.95**	**24.87**	**24.91**	**26.80**	**26.91**	**26.86**	**28.44**	**31.06**	**29.69**

Table 2: Ablation on datasets in our multi-stage pre-training processes for 1 run of experiments. The two sections show scores for different question types: SQL queries (top) and natural language (NL) questions (bottom). In a section each row shows a training process with different stages: Pre-training on Stage 2, pre-training on Stages 1+2, and all pre-training Stages 1+2+3. Table EM is table exact match accuracy; P is Precision; R is Recall; and F1 is F1 score for exact match of row, column, and cell.



(a) Table EM Accuracy (b) Row EM F1

(c) Column EM F1 (d) Cell EM F1

Figure 5: Evaluation results on Spider evaluation samples segregated by number of input tables.

correctness of individual table units without measuring the ordering. Column metrics measure predicted column *PetType* as correct and *avg(weight)* as incorrect without measuring ordering of the 2 columns. Row *cat* | *12.0* is measured as correct, while *dog* | *13.4* is measured incorrect without measuring the ordering among them. Out of the 4 target cells, *cat*, *dog*, *12.0* are measured as correct.

Impact of the number of input tables. The number of input tables increases the complexity of the questions and directly impacts the effectiveness of the models. We segregate evaluation on Spider validation set on the basis of number of input tables and compare the results to study the impact of input table number. We observe from Figure 5 that effectiveness reduces as the number of tables increases for both MultiTabQA and `tapex-base`. However, MultiTabQA fares better than `tapex-base` when the number of input tables increases. MultiTabQA generates whole tables, rows, columns and cells better than `tapex-base` as observed in Figure 5a, 5b, 5c and 5d. The gain of MultiTabQA in table exact match for one-table context is around 8.81%, for two-tables context around 4.37%, and it performs similar to `tapex-base` for three-tables context. It also has a significant higher score on rows, columns and cells, on both single and multi-tabular context.

We also observe that while the column and table EM decreases dramatically when using several tables (Figure 5a and 5c), the row and cell EM does not (Figure 5b and 5d). This indicates that MultiTabQA can generate rows and cells as effectively in single and multiple input tables settings but fail to do so for columns and consequently for the whole table. This is due to the fact that certain columns in the answer, particularly ones with numbers such as floats, are challenging to generate. The error from the incorrect columns propagates and are accumulated in the table EM leading to a significant drop in performance for multi-table queries.

Ablation on training stages. We perform ablation on the pre-training stages to analyse the contribution of each dataset. The simplest setting is to pre-train with Spider SQL queries, i.e., Stage 2. To evaluate the effectiveness of single table Tapex pre-training samples, the next setting comprises of stages 1 and 2, i.e., pre-train with Tapex pre-training and Spider SQL dataset. The final comparison is with the three-stage pre-training as described in Section 4.1. The results for one run of the experiments are displayed in Table 2. We observe that table exact match is highest for both pre-training and fine-tuning for the three-stage training. Stage 2 fares better than Stage 1+2 on table exact match,

and generally has better precision and F1 scores but lower recall. The three-stage pre-training with our synthetic data augmented with Spider outperforms the other settings and confirms the effectiveness of our synthetic data samples in boosting model efficacy.

7 Related work

Tabular QA is a research direction in the broader topic of table understanding (Jena et al., 2022; Shigarov, 2022) in natural language processing. Recent advances in table representation (Eisenschlos et al., 2021) and pre-training (Cheng et al., 2021a; Liu et al., 2022; Cheng et al., 2021a), table fact verficiation (Gu et al., 2022; Zhou et al., 2022b), table numeric reasoning (Shankarampeta et al., 2022; Zhou et al., 2022a), table-to-text generation (Andrejczuk et al., 2022), text-to-table generation (Wu et al., 2022), table summarization (Jain et al., 2018; Chen et al., 2013; Zhang et al., 2020), and table question answering (Yin et al., 2020; Zhang et al., 2020; Herzig et al., 2020; Zhu et al., 2021; Liu et al., 2021; Cheng et al., 2021b; Nan et al., 2021; Ma et al., 2022; Pal et al., 2022; Jin et al., 2022; Zhou et al., 2022a) have shown the adaptability of language models to table processing.

8 Conclusion

In this work, we propose a new task of multi-table question answering without intermediate logical forms to fill the gap of existing end-to-end table QA research which focused only on single-table QA. We release a pre-training dataset of $132,645$ samples to effectively train a seq2seq model. We fine-tune and evaluate our model, MultiTabQA, on natural language questions of three datasets: Spider, GeoQuery and Atis, to test the efficacy in a multi-table setting. As many multi-table questions result in tables, we train the model to generate tables. This necessitates table-specific metrics at various levels of granularity which we design to evaluate the effectiveness of our model. We demonstrate that such metrics is insightful in understanding model behavior. MultiTabQA outperforms existing state-of-the-art single table QA model fine-tuned to adapt to a multi-table QA setting.

9 Limitations

Our synthetic pre-training dataset was automatically generated from manual templates, which inspite of dataset creation scalability and low cost, may limit the diversity of the generated SQL queries. Our model, MultiTabQA, requires improvement in numeracy understanding and numeric operations. Real numbers are especially challenging and the model may not be able to correctly generate all the digits of the number correctly rending the generated cell incorrect. Furthermore, large input tables pose a challenge as the input sequence may get truncated beyond the model's maximum sequence length. This has practical limitation in the size and number of input tables which the model can accommodate before truncation which leads to incorrect answers.

10 Ethical Considerations

The task and model proposed in the paper is aimed at broadening the scope of TabularQA research. All the datasets used in this research, apart from our synthetic data, are publicly available in peer-reviewed articles and referenced in this paper. The synthetic SQL dataset we release was generated over a standard benchmark database which has been annotated by 11 Yale students as mentioned in the original paper. Our synthetic samples use templates annotated by the authors of this work and do not use any user-specific data or information. We will be providing open access to our datasets for use in future research under the MIT License. All datasets, including the synthetic pre-training dataset and all datasets adapted for multi-table QA will be released. Our model is built over `tapex-base` which in turn has been trained over `bart-base`. Our work did not explicitly handle any bias which exists in the aforementioned pre-trained models.

11 Acknowledgements

We thank Elsevier's Discovery Lab for their support throughout this project and funding this work. This work was also supported by the Dutch Research Council (NWO) under project numbers 016.Vidi.189.039 and 314-99-301, by H2020-EU.3.4. Societal Challenges, Smart, Green and Integrated Transport (814961), and by the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through NWO, https://hybrid-intelligence-centre.nl. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

Ewa Andrejczuk, Julian Martin Eisenschlos, Francesco Piccinno, Syrine Krichene, and Yasemin Altun. 2022. Table-to-text generation and pre-training with TabT5. *arXiv preprint arXiv:2210.09162*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, page 41–48. Association for Computing Machinery.

Zefeng Cai, Xiangyu Li, Binyuan Hui, Min Yang, Bowen Li, Binhua Li, Zhen Cao, Weijie Li, Fei Huang, Luo Si, and Yongbin Li. 2022. Star: Sql guided pre-training for context-dependent text-to-sql parsing. *arXiv preprint arXiv:2210.11888*.

Jieying Chen, Jia-Yu Pan, Christos Faloutsos, and Spiros Papadimitriou. 2013. TSum: Fast, principled table summarization. In *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*. Association for Computing Machinery.

Zhoujun Cheng, Haoyu Dong, Fan Cheng, Ran Jia, Pengfei Wu, Shi Han, and Dongmei Zhang. 2021a. Fortap: Using formulas for numerical-reasoning-aware table pretraining. *arXiv preprint arXiv:2109.07323*.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2021b. HiTab: A hierarchical table dataset for question answering and natural language generation. *arXiv preprint arXiv:2108.06712*.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Julian Martin Eisenschlos, Maharshi Gor, Thomas Müller, and William W. Cohen. 2021. Mate: Multi-view attention for table transformer efficiency. In *Conference on Empirical Methods in Natural Language Processing*, page 7606–7619. Association for Computational Linguistics.

Zihui Gu, Ju Fan, Nan Tang, Preslav Nakov, Xiaoman Zhao, and Xiaoyong Du. 2022. Pasta: Table-operations aware fact verification via sentence-table cloze pre-training. *arXiv preprint arXiv:2211.02816*.

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.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 963–973. Association for Computational Linguistics.

Parag Jain, Anirban Laha, Karthik Sankaranarayanan, Preksha Nema, Mitesh M. Khapra, and Shreyas Shetty. 2018. A mixed hierarchical attention based encoder-decoder approach for standard table summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 622–627. Association for Computational Linguistics.

Aashna Jena, Vivek Gupta, Manish Shrivastava, and Julian Eisenschlos. 2022. Leveraging data recasting to enhance tabular reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4512 – 4525. Association for Computational Linguistics.

Nengzheng Jin, Joanna Siebert, Dongfang Li, and Qingcai Chen. 2022. A survey on table question answering: Recent advances. *arXiv preprint arXiv:2207.05270*.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. KaggleDBQA: Realistic evaluation of Text-to-SQL parsers. *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)*, page 2261–2273.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian guang Lou. 2021. TAPEX: Table pre-training via learning a neural SQL executor. *arXiv preprint arXiv:2107.07653*.

Ruixue Liu, Shaozu Yuan, Aijun Dai, Lei Shen, Tiangang Zhu, and Meng Chen. 2022. Few-shot table understanding: A benchmark dataset and pre-training baseline. In *Proceedings of the 29th International Conference on Computational Linguistics*, page 3741–3752.

Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2022. Open domain question answering with a unified knowledge interface. In *Annual Meeting of the Association for Computational Linguistics*, pages 1605–1620. Association for Computational Linguistics.

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. *arXiv preprint arXiv:2104.00369*.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2021. Investigating the limitations of transformers with simple arithmetic tasks. *arXiv preprint arXiv:2102.13019*.

Kuntal Kumar Pal and Chitta Baral. 2021. Investigating numeracy learning ability of a text-to-text transfer model. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, page 3095–3101. Association for Computational Linguistics.

Vaishali Pal, Evangelos Kanoulas, and Maarten Rijke. 2022. Parameter-efficient abstractive question answering over tables or text. In *Proceedings of the Second DialDoc Workshop on Document-grounded Dialogue and Conversational Question Answering*, pages 41–53. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

P. J. Price. 1990. Evaluation of spoken language systems: the ATIS domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Abhilash Shankarampeta, Vivek Gupta, and Shuo Zhang. 2022. Enhancing tabular reasoning with pattern exploiting training. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*, page 706–726. Association for Computational Linguistics.

Tianze Shi, Chen Zhao, Jordan Boyd-Graber, Hal Daumé III, and Lillian Lee. 2020. On the potential of lexico-logical alignments for semantic parsing to SQL queries. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, page 1849–1864. Association for Computational Linguistic.

Alexey O. Shigarov. 2022. Table understanding: Problem overview. *WIREs Data Mining and Knowledge Discovery*, 13:e1482.

Xueqing Wu, Jiacheng Zhang, and Hang Li. 2022. Text-to-Table: A new way of information extraction. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2518–2533.

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

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2021. GraPPa: Grammar-augmented pre-training for table semantic parsing. In *International Conference on Learning Representations*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, page 1050–1055. AAAI Press.

Shuo Zhang, Zhuyun Dai, Krisztian Balog, and Jamie Callan. 2020. Summarizing and exploring tabular data in conversational search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1537–1540. ACM.

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

Fan Zhou, Mengkang Hu, Haoyu Dong, Zhoujun Cheng, Shi Han, and Dongmei Zhang. 2022a. TaCube: Pre-computing data cubes for answering numerical-reasoning questions over tabular data. *arXiv preprint arXiv:2205.12682*.

Yuxuan Zhou, Xien Liu, Kaiyin Zhou, and Ji Wu. 2022b. Table-based fact verification with self-adaptive mixture of experts. *arXiv preprint arXiv:2204.08753*.

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. *arXiv preprint arXiv:2105.07624*.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
9

☑ A2. Did you discuss any potential risks of your work?
10

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

3

☑ B1. Did you cite the creators of artifacts you used?
3

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
3

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
3

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Dataset is standard benchmark semantic parsing dataset publicly available for more than a decade.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
3

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
3

C ☑ Did you run computational experiments?

6

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
6

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
6

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
6

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
3

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.