Table-To-Text generation and pre-training with TABT5

Anonymous ACL submission

Abstract

Encoder-only transformer models have been successfully applied to different table understanding tasks, as in TAPAS (Herzig et al., 2020). A major limitation of these architectures is that they are constrained to classification-like tasks such as cell selection or entailment detection. We present TABT5, an encoder-decoder model that generates natural language text based on tables and textual inputs. TABT5 overcomes the encoder-only limitation by incorporating a decoder component and leverages the input structure with table specific embeddings and pre-training. TABT5 achieves new state-of-the-art results on several domains, including spreadsheet formula prediction with a 15% increase in sequence accuracy, QA with a 2.5% increase in sequence accuracy and data-to-text generation with a 2.5% increase in BLEU.

1 Introduction

Large language models (LLMs) such as BERT (Devlin et al., 2019) or T5 (Raffel et al., 2020) have shown impressive abilities to encode and generate fluent and coherent natural language text (Lan et al., 2020; Gururangan et al., 2020; Conneau et al., 2020). However their representation and generational capabilities are limited when it comes to structured or semi-structured domains like tables. This is mainly due to two reasons: (i) LLMs are only pre-trained on large amount of unstructured data (e.g., documents, news, etc.); (ii) their underlying model architecture lacks a way to fully leverage this structure information.

Yet, structured and semi-structured data is ubiquitous on the web (e.g. web tables, database tables, PDF tables, spreadsheets store rich numerical information and provide concise summaries of data), and widely studied in the academia (Chen et al., 2021; Cheng et al., 2022; Parikh et al., 2020; Wang et al., 2021) and the industry (e.g. formula prediction in Excel\footnote{https://www.microsoft.com/excel} and Google Sheets\footnote{https://www.google.com/sheets/about}, or extracting data from tables in Text-to-Speech Assistants).

Recently, several solutions propose to alleviate aforementioned issue by introducing pre-training or intermediate training strategies for tables. For instance, Herzig et al. (2020) propose to use a Masked Language Model (MLM) as pre-training objective to improve the contextual representation of BERT (Devlin et al., 2019) over table inputs. To train their model, they introduce additional input embeddings that help the model understand the table structure. These pre-training models are designed and evaluated on datasets where the answers contain only table cells or aggregations of multiple cells, and not full sentences. In this paper, we tackle a set of distinct, complex tasks such as question answering and formula prediction that require full generation capabilities.

In particular, our contributions are as follows:

- We present an encoder-decoder based model TABT5 (Table-and-Text-to-Text Transfer Transformer) that can be applied to data-to-text generation tasks by relying on special embeddings of the input structure.
- We introduce different pre-training strategies that leverage web data containing tables.
- We evaluate our approach on three different table and text datasets in English and obtain state-of-the-art performance on several domains.

2 Problem Definition

The objective of our model is to learn a conditional sequence generator $P(y|x)$ where $x$ is endowed with extra two-dimensional structure. To encode said structure, each instance of $x$ is as a variable length sequence of tuples $(u_i, t_i, c_i, r_i)_{i=1}^N$ representing the components of $x$. In each component, $u_i$ is a natural language utterance, $t_i$ is the discrete
Figure 1: TabT5 linearizes the input table row by row and adds column and row embeddings to encode the 2-dimensional coordinates of each cell. The model is pretrained by randomly replacing 15% of cells by a \textless X\textgreater marker and training the decoder to predict the hidden output in sequence.

3 Related Work

Table language models. Several works use a common serialization where table contents are linearized row by row (Parikh et al., 2020; Wang et al., 2021; Iida et al., 2021; Eisenschlos et al., 2021; Herzig et al., 2020). Another design choice is to use structural positional encoding, in addition to 1-D encoding, to represent two dimensional information such as the row and column positions of tokens (Herzig et al., 2020; Eisenschlos et al., 2021; Iida et al., 2021; Wang et al., 2021). An alternative is the use of a structure-aware attention, in contrast to a standard self-attention mechanism, to better leverage the table structure (Mueller et al., 2019; Yang et al., 2022). All of these models are encoder-only. Concurrent with our work Shi et al. (2022) propose a similar method to adapt T5 to tabular data, however their pretraining approach relies on existing annotated datasets and focuses solely on QA applications.

Table Pre-training. Most pretraining methods follow the Masked Language Modeling (MLM) scheme, where some percentage of input tokens are randomly masked and successively predicted in an encoder only setup (Herzig et al., 2020; Eisenschlos et al., 2021; Yang et al., 2022). Some approaches (Wang et al., 2021; Yin et al., 2020; Iida et al., 2021) apply the masking on a cell-level, where the full contents of a given cell is masked and then predicted. Our work differs in training the encoder and decoder jointly by using a de-noising scheme similar to the one used in T5 (Raffel et al., 2020).

Table QA. Given an input table, the task consists in producing an answer to a natural language question. We focus on WikISQL (Zhong et al., 2017), and learn an encoder-decoder model with row/column embeddings in the weakly supervised setting without logical forms. Herzig et al. (2020) use a similar approach with a BERT encoder-only model, while Liu et al. (2022) use a BART encoder-decoder model without extra embeddings.

Formula prediction. The task is to predict formula conditioned on headers and other contextual information, without an explicit natural language question. Chen et al. (2021) propose to use a BERT-based architecture to compute an input header and a cell data vector that are fed to a two-step LSTM decoder. The decoder proposes a formula sketch and refines it with cell ranges. Cheng et al. (2022) propose a similar approach where the representation of the target cell output by a table encoder (Wang et al., 2021) is an input to a two-step LSTM-based decoder. Our approach is simpler as a single model is used to solve the task end to end.

Data-to-Text. The task consists in generating a natural language description given structured data input. Parikh et al. (2020) employ an encoder-decoder model where the encoder and decoder are both initialized with BERT (Devlin et al., 2019). Kale and Rastogi (2020) use a T5 model. In both, tables are linearized with row/column separator tokens. Our work differs as we use row/column embeddings, and we employ two pretraining schemes.

4 TabT5 Model

TabT5 uses the T5 pre-trained language model as a baseline architecture. We linearize the table into a sequence of words, split words into word pieces (tokens) and concatenate the question and table tokens to create the input sequence. We include in the model row and column embeddings to encode table structure (Herzig et al., 2020). We add them on top of the token embeddings as model inputs and op-
timize them during training (Figure 1). The target sequence is a free-form answer. This can be an answer to a question for question-answering tasks, a table summary when no question is specified or a formula for the formula prediction tasks.

5 Pre-training

As a starting point, we use publicly available T5 checkpoints released by Raffel et al. (2020). Next, we pre-train TABT5 on Wikipedia tables. We use the pre-training dataset proposed by Herzig et al. (2020) which contains 6.2M tables (3.3M of class Infobox and 2.9M of class WikiTable). We also extract related passages that caption the table. We define two pre-training strategies described below.

5.1 Denoising

We design a denoising strategy for table-like data, following the method used in T5 (Raffel et al., 2020), by training the model to predict a target sequence containing the missing or corrupted tokens in the input table. The target consists of all of the dropped-out spans of tokens, delimited by the sentinel token used in the input sequence (Figure 1). We replace 15% of table cells and columns in the input with a mask token. This helps the model capture relationships between the neighbouring cells and between the related text.

5.2 ToTTification

We another pre-training strategy using the same Wikipedia tables (Section 5.1) inspired by ToTTTo (Parikh et al., 2020), to be used after denoising. For each table, we retrieve the statements that are in the same page as the table or link to the table page. We only keep statements that have an entity (Wikipedia URL, number or date) that matches the table, 4M in total. These statements become our target text. We add the matching entities in those statements as a (comma separated) plain text component of the input to guide the generation.

6 Experiments

6.1 Datasets

WikiSQL (Zhong et al., 2017) is a Table-QA dataset containing 80,654 instances. To create the dataset, crowd workers paraphrase a template-based question into natural language. Two other crowd workers’ groups then verify and correct the quality of the proposed paraphrases. We follow the approach of Herzig et al. (2020) and generate the reference answer from the reference SQL provided using our own SQL implementation.

ENRON (Chen et al., 2021) is a dataset to evaluate formula prediction task containing over 17K spreadsheets extracted from the Enron email corpus that contains 218,798 instances. It focuses on formula with referenced cells in a rectangular neighbourhood region of the target cell and the headers. We preprocess the data as described Appendix C.

TOTTO (Parikh et al., 2020) is a Table-to-Text dataset containing 120,761 instances. It consists of tables paired with table-grounded sentences as natural language descriptions. Parikh et al. apply several heuristics to sample tables and candidate sentences from Wikipedia pages. They use crowd worker annotators to highlight the corresponding table cells and revise natural language descriptions.

6.2 Results

We discuss the experimental setup in the Appendix B. For ToTTTo, we report the results in Table 1. We follow Parikh et al.’s official script to compute BLEU and PARENT as the evaluation metrics. The Non-Overlap dev set features examples that are out-of-domain from the training set. For the test set, we provide results from one run as this is a laborious manual process requiring a submission of test files into an external source. Note that Parikh et al. do not provide development set results in their paper and Kale and Rastogi do not provide test set results for the base model. We observe that TABT5 outperforms SOTA models and its performance is improved further by using the TOTTIFY pre-training. Note that the base model performs slightly better than the large model. We believe that the large model requires more careful hyperparameters tuning to achieve higher results. For WikiSQL and ENRON, results are reported in Table 2 and Table 3 respectively. Also, see the Appendix C for additional results on the ENRON dataset. We observe that TABT5 significantly improves over SOTA performance for both WikiSQL (> 30% of error reduction in the base variant) and ENRON (35% error reduction in the base variant).

Note that TABT5 in the base variant (220M parameters) outperforms other models with substantially

---


4Raffel et al. experimentally show that 15% corruption rate works best. We use the same rate for our denoising objective.

6The details on submissions for the ToTTTo test set can be found in https://github.com/google-research-datasets/ToTTTo
higher number of parameters (e.g. BERT used in Herzig et al. (2020) has 380M parameters and BART_{large} in Liu et al. (2022) ~ 418M). Additionally, TABT5 in the small variant (60M parameters) achieves high accuracy compared to SOTA for the ENRON dataset. When increasing the model size, we observe an increase in performance for both datasets. For WikiSQL the large variant (770M parameters) achieves exceptionally high sequence accuracy of 95% (53% error reduction wrt. to the baseline performance).

Table 2: Table-QA results on WikiSQL in the weakly supervised setting without logical forms. TABT5 provides gains over existing approaches even in a small model variant. The large model gives the best results.

**Ablation** We perform the ablation study only on the base variant due to computational costs. For each experiment, we report two ablations runs: (i) - denoising indicates that we remove the denoising pre-training, and (ii) - embeddings indicates runs without row and column embeddings. We observe that the performance of TABT5 deteriorates when removing either denoising or embeddings. This

---

(The gap between T5-base and Kale and Rastogi comes from using distinct versions of T5. We reproduced their results using v1.0. We use v1.1 with the same set of hyperparameters.)

Table 3: Formula prediction results on ENRON. The T5-base baseline brings substantial improvements over existing approaches. TABT5 provides further gains, with the large model variant obtaining the best results.

**Error analysis** We manually annotate a random sample of 80 errors made by TABT5. We find that 55% are paraphrases and 72.5% overall are acceptable (correct content with some details missing). We classify the remaining errors into grammatical and data-to-text generation. In future work, we show they are crucial for all tasks. We also show results for the TOTTification, which improves the performance for TABT5 but it is detrimental for the other tasks, compared to the denoising method.

---

7 Conclusions and Discussion

We introduced TABT5, a new T5-based encoder-decoder model that achieves new SOTA results on spreadsheet formula prediction, question answering and data-to-text generation. In future work, we plan to use larger and task specific datasets for pre-training (e.g. scrape tables from Web, sheets).
Ethical Considerations

As is true for existing works on generative architectures based on large language models, there are potential risks and harms associated with using the output for downstream applications (Bender and Koller, 2020; Brown et al., 2020). Beyond the original pre-trained checkpoint from T5, we also used tables from Wikipedia for intermediate pre-training, which may contain additional undesirable biases.

References


A Appendix A. Hyperparameters Selection

We run denoising pre-training for 1M steps and ToTTify pre-training for 100k steps on top of denoising. We set each fine-tuning task for 50k training steps. We run the evaluation on ENRON and WIKI-SQL using the default T5 hyper-parameters with an input sequence length of 1024 and output 256. For the ToTTO dataset, we follow the approach of Kale and Rastogi (2020) and keep the learning rate constant and equal to $1 \times 10^{-4}$, an input and output sequence length is equal to 512 and 256 respectively, and batch size is 256. Additionally, we observe that TABT5 in the small and base variants overfit quickly. Thus, we decide to increase the dropout rate to 0.2 when using pre-training.

B Appendix B. Experimental Setup

We apply the standard T5 tokenizer and start pre-training from publicly available T5 checkpoints. Row and column embeddings are randomly initialized. We run pre-training and fine-tuning on a setup of 16 Cloud TPU v3 cores with maximum sequence length of 1024. Pre-training takes around 3, 8 and 13 days for small, base and large models. Fine-tuning takes around 2 – 3 hours for each task. For each dataset, we run five independent runs and report median and standard deviation.

C Appendix C. ENRON results.

In this section, we present the results on the ENRON dataset that contains all original data (i.e. all formulas in the tables). We find these results interesting as the ENRON contains real data collected by the company. Thus, we believe this scenario is realistic. We present the results in Table 4. We observe that our results are extremely high because in ENRON dataset over 70% of tables contain a target formula in the input table. Following the previous approaches, we make the task harder. In the experimental section of the paper, we preprocess the dataset by removing all formulas from the input table cells. Additionally, we remove examples containing (i) erroneous formulas, and (ii) ranges from different tables in both input tables and target formulas.

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>T5-base</td>
<td>93.05 ± 0.98</td>
</tr>
<tr>
<td>TABT5-small</td>
<td>95.39 ± 0.17</td>
</tr>
<tr>
<td>TABT5-base</td>
<td><strong>95.59 ± 0.08</strong></td>
</tr>
<tr>
<td>+TOTTIFY</td>
<td>95.50 ± 0.05</td>
</tr>
<tr>
<td>-DENOISING</td>
<td>93.92 ± 0.16</td>
</tr>
<tr>
<td>-EMBEDDINGS</td>
<td>95.00 ± 0.16</td>
</tr>
</tbody>
</table>

Table 4: Formula prediction results on ENRON. In this experiment, the model has to produce the target formula having access to the formula used in the surrounding cells. Results are higher wrt. to Table 3 as the model is allowed to “copy” already used formulae or part of them.

D Appendix D. Error analysis

We manually annotate 80 errors made by the TABT5 and classify them in Figure 2. 35% of the TABT5 output are exactly the same as T5’s output where 55% are correct (paraphrases) and 72.5% acceptable answers.
Figure 2: We manually annotate 80 errors made by TABT5. We find that 55% of predictions are paraphrases and 72.5% are acceptable. The classification of error types is given in Table 5.

<table>
<thead>
<tr>
<th>Error type</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paraphrase</td>
<td>Express the same meaning as the ground truth using either synonyms or the exact words in a different order.</td>
<td>TABT5 output: Alma Jodorowsky played Evelyn in Kids in Love. Ground Truth: Alma Jodorowsky had the role of Evelyn in 2016 film Kids in Love.</td>
</tr>
<tr>
<td>Acceptable missing information</td>
<td>The content is correct, but it is missing some details that do not affect the answer’s meaning.</td>
<td>TABT5 output: The 500 Questions was aired in Germany on RTL from July 4 to August 14, hosted by Günther Jauch. (year is missing) Ground Truth: In 2016, RTL television aired 500-DQA in Germany and was hosted by Gunther Jauch</td>
</tr>
<tr>
<td>Wrong missing important information</td>
<td>The content is missing some details that affect the meaning of the answer or are essential for understanding the answer.</td>
<td>TABT5 output: Putney railway station is in the Wandsworth borough and is in Zone 2. (missing zone 3 could be important information) Ground Truth: Putney railway station serves Putney in the London borough of Wandsworth in southwest London and in zones 2 and 3</td>
</tr>
<tr>
<td>Hallucination</td>
<td>Intrinsic – The generated output contradicts the source content – Extrinsic – The generated output cannot be verified from the source content –</td>
<td>TABT5 output: In 1924, William Glackens received the Temple Gold Medal for his work “Natural form”. (We cannot verify the work’s name) Ground Truth: William Glackens won the 1924 award from Temple Gold Medal Nude.</td>
</tr>
<tr>
<td>Grammatical errors</td>
<td>The sentence is grammatically incorrect</td>
<td>TABT5 output: As of the census of 2000, there were 42,695 people residing in the Watauga County. Ground Truth: As of the census of 2000, there were 42,695 people residing in Watauga county.</td>
</tr>
<tr>
<td>Wrong</td>
<td>Other errors such as: wrong aggregation (counts, sums, etc), swapped arguments that change the meaning of the sentence.</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Error types definition and examples.