Improving Factual Accuracy of Neural Table-to-Text Output by Addressing Input Problems in ToTTo

Barkavi Sundararajan and Somayajulu Sripada and Ehud Reiter
Department of Computing Science, University of Aberdeen
{b.sundararajan.21, yaji.sripada, e.reiter}@abdn.ac.uk

Abstract

Neural Table-to-Text models tend to hallucinate, producing texts that contain factual errors. We investigate whether such errors in the output can be traced back to problems with the input. We manually annotated 1,837 texts generated by multiple models in the politics domain of the ToTTo dataset. We identify the input problems that are responsible for many output errors and show that fixing these inputs reduces factual errors by between 52% and 76% (depending on the model). In addition, we observe that models struggle in processing tabular inputs that are structured in a non-standard way, particularly when the input lacks distinct row and column values or when the column headers are not correctly mapped to corresponding values.

1 Introduction

Table-to-Text generation refers to the task of generating natural language descriptions from tabular data (Parikh et al., 2020; Chen et al., 2020a,b) and is widely used in several application domains such as medical diagnosis (Pauws et al., 2018), financial (Zhu et al., 2023) and weather reporting (Sripada et al., 2002; Gkatzia et al., 2017; Upadhyay and Massie, 2022) and sports summaries (Thomson et al., 2020). Neural language models are known to generate fluent texts (Ji et al., 2023) but may generate outputs that are factually incorrect or unrelated to the provided input data. Such undesirable generation is called ‘hallucination’ (Wang and Sennrich, 2020; Raunak et al., 2021; Ji et al., 2023).

Previous studies on Table-to-Text tasks adopt traditional seq2seq methods to generate table descriptions (Wiseman et al., 2017; Puduppully et al., 2019; Rebuffel et al., 2019). Recently, Transformer based models (Devlin et al., 2019; Raffel et al., 2020; OpenAI, 2023) have shown remarkable progress in language generation from textual input (Badaro et al., 2023), however tabular data still needs more improvement to control hallucinations (Rebuffel et al., 2021). Neural models struggle with tabular data, especially when the inputs do not have distinct cell values from rows and columns mapped along with their respective headers. These input problems lead the model to generate more factual errors (Kasner and Dušek, 2024).

Using the ToTTo tabular dataset (Parikh et al., 2020), we identify and address input problems that are responsible for factual errors. Some common tabular input problems in the ToTTo are i. ‘non-atomic’ cell values, where a column contains multiple values such as leader name, party name and % of votes in one cell rather than a single indivisible value, ii. missing important cell values in the input (see Table 1) and iii. nested column headers and row headers in the Wikipedia tables that lead to incorrect mapping of the cell values.

Table 1 presents a sample from ToTTo. Only the highlighted cells from Table 1a are passed to the model (as shown in Table 2). Passing Norton’s % of votes and her party name (compare Table 1b to Table 1a) eliminates the hallucinated % of votes (see output for Table 1b); this correction is based on the input problem described in Sec. 5.2.

In this paper, we score the quality of output texts by manually annotating output errors instead of using automatic evaluation metric scores such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), PARENT (Dhingra et al., 2019) and BLEURT (Selman et al., 2020). We conducted a pilot study, where we fine-tuned T5-base and T5-large models (Raffel et al., 2020), analysing 1,677 politics domain texts from the ToTTo dataset through manual error analysis adopted from Thomson and Reiter (2020). These manual error annotations allowed us to identify patterns of errors in the generated text which were then traced back to input problems.

Our approach is summarised as follows:

I. We systematically correct the tabular inputs for the politics domain in ToTTo to adhere to a standard form to ensure the generation of factual texts from neural models. The correction procedure is elaborated in Sec. 5.2 and is supplemented by pseudocode in App. D.

(a) We apply this correction to a larger subset of 210 samples, resulting in a 62% decrease in factual errors for T5-base and a 57% decrease for T5-large in the generated text (Sec. 6.1).

(b) We conduct experiments on Llama 2-7B and Llama 2-13B models (Touvron et al., 2023) with 40 challenging samples selected from the previous 210 samples. Tailoring zero-shot prompts for specific input and error annotation on 160 texts showed that correcting input reduces factual errors by 52% in Llama 2-7B and 76% in Llama 2-13B (Sec. 6.2).

II. The manual error annotation methodology adopted from Thomson and Reiter (2020) is detailed in App. B; this builds on the work of Sundararajan et al. (2022) for ToTTo politics domain outputs. The inter-annotation agreement on the error annotation was good, with a Fleiss’ Kappa of 0.622 (Sec. 7).
BLEU (Papineni et al., 2002), PARENT (Dhingra et al., 2019) and BLEURT (Sellam et al., 2020) by few points in the leaderboard, the fundamental problem with the tabular input remains.

Chen et al. (2022) acknowledged that the target cells in the ToTTo tabular input are not always highlighted in their work titled ‘Table Structure Understanding and Text Deliberation Approach’. They used a template to extract all facts from the raw table for 1.2K training samples (only for the inputs with rows and columns fewer than 8), and employed hierarchical multi-head attention to capture the structural information in their fine-tuning process. Though this approach promises to retain the facts from raw tables, it only addresses simpler tables with fewer than 8 rows and columns, still has limitations for longer tables, complex tabular structures and non-atomic tabular cells.

Our focus on correcting inputs aiming to achieve factually correct outputs aligns with the work of Dušek et al. (2019) on the E2E dataset (Novikova et al., 2017). Their study also demonstrated that improving inputs in an NLG dataset helps in improving model outputs.

**Error Analysis:** While the automatic metric scores such as BLEU, PARENT and BLEURT help evaluate the model’s performance at a high level, relying solely on these metrics will not address specific weaknesses i.e., lower metric scores do not provide insights into specific error types in the output. We follow guidelines from van Miltenburg et al. (2021) to perform error analysis in NLG systems and investigate errors in the output at a more granular level by adopting the manual error annotation approach from Thomson and Reiter (2020); Sundararajan et al. (2022); Thomson et al. (2023).

Maynez et al. (2020) also emphasized automatic metrics are not sufficient to study the hallucination problem and provided a detailed study on intrinsic and extrinsic hallucination in abstractive summarization. We studied hallucination in our evaluation scheme by annotating different categories of errors in the output tokens (single token or group of tokens). Mapping our adopted methodology to Maynez et al. (2020)’s work, intrinsic is the main error category occurring when generated outputs are not faithful to the given input. It includes **WORDS** with **NAME** and **DATE**

**CONTEXT** and **OTHER** categories. Extrinsic refers to our **ADDITION** category (see App. B.3).

**LLM prompts:** Empirical evaluation of prompting strategies on the three large language models (LLMs) by Sivarajkumar et al. (2023) in clinical NLP found that tailoring task-specific prompt is crucial for achieving accuracy. In a study on Text-to-SQL, Chang and Fosler-Lussier (2023) investigated zero-shot prompting strategies, highlighting the significance of table representation. Their findings indicated that normalized database prompts outperformed unnormalized ones; this motivates our initial step of correcting tabular inputs to a standard form. In our work, we leveraged a recent LLM, LLAMA 2 (Touvron et al., 2023) and tailored our zero-shot prompt (Kojima et al., 2022) specific to the content of each table.

## 3 Pilot Study

### 3.1 Methodology

We only look at the **politics** domain on the ToTTo validation set. We build upon the work of Sundararajan et al. (2022) to identify the causes of output errors in T5 models. In this paper, we go beyond error annotations to fix these errors in our main study, both in T5 and LLAMA 2 models (detailed in Sec. 5).

The error categories mentioned in Sundararajan et al. (2022) are: **WORD**, **NAME**, **DATE**

**CONTEXT**, **NOT**

**CHECKABLE** and **NON-ENGLISH**. In this work, we excluded **NOT**

**CHECKABLE** and introduced a new error category, **ADDITION**, which is used when the generated text has added words or phrases that diverge from the input. Definitions for all error categories are provided in App. B.3.

### 3.2 Insights from our Pilot Study

For the pilot study, we fine-tuned both the T5-base (T5-b) and the T5-large (T5-l) models on the ToTTo dataset by following the baseline approach (Kale and Rastogi, 2020). The hyperparameters and fine-tuning details for these two models are shown in App. A.

Our analysis (presented in Table 3) shows that:

**No Error:** 47% of the samples from T5-b and 60% of the samples from T5-l are error-free.

**Omissions:** Omissions occur when the generated text fails to mention some information from
the input (González Corbelle et al., 2022) without making any factual errors. If the output has errors and omissions, we classify it as an error. The T5-b had 36% omissions and T5-l had 29%.

**Errors:** Our analysis revealed that T5-b made factual errors in 17% and T5-l made factual errors in 11% of the total samples.

**Hypothesis:** Based on the insights from this analysis, we hypothesize that when tabular input data is structured in non-standard ways, models struggle to interpret these ambiguous inputs leading to generate factually inaccurate output. We test this hypothesis by addressing input problems related to non-standard tabular structures.

## 4 Input Problems

Due to the practical challenges involved in improving the tabular input for the entire dataset, which includes unique headers and tabular structures for each input, our focus is on analyzing a subset of samples containing errors. We examined 124 error samples from the T5-b and 86 error samples from T5-l models within the ToTTo politics domain, as identified in Table 3. We aim to segregate the errors originating from non-standard or illogical nested table structures. We categorize these input problems into two broad categories, as briefly elaborated upon in Sec. 4.1 and Sec. 4.2.

### 4.1 Generic Input Problems

**Non-atomic tabular cell values:** When a table cell contains multiple atomic values (see Table 4). Examples of such non-atomic forms include multiple leaders’ names, votes, term dates, or election years all in a single cell. We further categorize these problems into ‘single record lacking atomicity’ and ‘multiple records lacking atomicity’ (shown in Fig. 2 in App. F) to demonstrate how the models struggle when records of multiple leaders lack atomicity.

**Complex table type:** When a table contains election results in sentence form, models struggle to interpret and generate meaningful texts because the sentence form data lacks the needed context (see Table 11 in App. F).

**Insufficient input:** In some cases, the necessary cells are not highlighted in the tabular input, resulting in incorrect outputs. Our analyses in Table 1 and Table 5 demonstrate that outputs become factually correct when relevant cells are included.

**Longer table input:** Models often struggle to generate accurate texts for lengthy table inputs, especially when the data is not in a standard form and lacks clear cell relationships.

### 4.2 Politics Domain-Specific Input Problems

**Politics specific headers:** In ToTTo, the use of symbols, for example, ‘+’ or ‘-’ instead of having clear semantic terms like ‘swing percentage’ or ‘% change compared to previous election’ as column headers caused 5% of errors. This lack of semantic guidance in the input made it difficult for models to accurately generate the correct output text (see Table 10 in App. F).

**List of leader names in the input:** In the politics domain, we observed a specific issue when input data contains a list of leader names. Models tend to favour the leader whose name appears first in the list (for example, either from the table title or as the first leader in highlighted cells), even if they have lost the election (see Table 13, Table 14, Table 15 in App. F). This becomes even more challenging when leader names are associated either with missing values (opponent leader’s name or vote count) in other columns in the input or when the tabular cell values are non-atomic.

The manual fixes we applied to each of these input problems with examples are detailed in Sec. 5.2.

## 5 Main Study

### 5.1 LLAMA 2 Models

In our main study, in addition to T5 models, we included LLAMA 2-7B (L-7B) and LLAMA 2-13B (L-13B) models to test our hypothesis on non-standard tabular inputs. We received official model weights from Meta\(^4\) and then quantized our models by following the installation instructions from https://github.com/ggerganov/llama.cpp.

---

\(^4\) We downloaded weights from [https://ai.meta.com/resources/models-and-libraries/llama-downloads/](https://ai.meta.com/resources/models-and-libraries/llama-downloads/) and quantized our models by following the installation instructions from [https://github.com/ggerganov/llama.cpp](https://github.com/ggerganov/llama.cpp).
In the United States Senate elections of 2014, Dan Sullivan (Republican) won in Alaska with 48.0% of votes against Mark Begich (Democratic) who got 45.8%. In Virginia, Mark Warner (Democratic) won with 49.1% of votes against Ed Gillespie (Republican) who got 48.3%.

LLAMA 2 models. We ran experiments using the full model weights and the model with 4-bit integer quantization for the ToTTo dataset (politics domain). Both produced comparable output quality with no difference in errors for the quantized models. Therefore, we finalized our experiments with 4-bit quantized models to save computational resources and used them for local inference on a MacOS CPU. We set the temperature to 0.0 as it produced fewer errors. Further details on data corrections for these models are discussed in Sec. 5.4.

5.2 Manual Fixes for Input Problems

We followed a systematic procedure to apply manual fixes to specific input problems, as discussed from Sec. 5.2.1 to Sec. 5.2.6. A supplementary pseudocode for these fixes is also provided in App. D. The algorithm 1 takes tabular data and title (metadata) from ToTTo as input parameters to execute six functions to gather insights and return with corrected tabular data. The first two functions, DATASIZEMANAGEABLEFORLONGTABLE and IDENTIFYLEADERNAMEORDER do not correct the tabular input but provide insights on the input problems leading to factual errors in the outputs. The other four functions correct the input problems as shown in algorithm 4, algorithm 5, algorithm 6 and algorithm 7.

5.2.1 Non-Atomic Cell Values

Correction: We corrected the input to store individual leader data, including votes, party, and election details, as a separate column variable. In Table 4a, when the ‘candidates’ column is not atomic, all models made errors for this tabular data. The corrections involved separating each leader’s party and votes into separate columns (see Table 4b). We passed the leading two leaders’ data, excluding the remaining leaders, in separate rows for the respective states (Alaska and Virginia). After input correction, T5-b and L-7B outputs in Table 4 were error-free. While some models omitted other candidate and election details, all models corrected the previous factual errors after the correction.

5.2.2 Complex Table Type

Correction: Complex tabular structures were generally difficult to correct because the election results are in sentence form e.g., see ‘Results’ column in Table 11 in App. F. We could only make corrections to the ‘non-atomic’ cell values in ‘candidates’ column. Three models generated “the incumbent senator Coe I. Crawford lost renomination to Ed- win S. Johnson, a Democrat candidate”. Losing renomination is when the current senator did not lose the seat to the opponent, but rather failed to be nominated in the primary to stand for reelection. This resulted in CONTEXT error, specifically when the models made unsupported assumptions.
Table 5: Fixing Insufficient input problems, as detailed in Sec. 5.2.3. ‘Before’ shows WORD and CONTEXT errors due to missing choices of votes. ‘After’ shows these errors are resolved when these cell values are provided.

about the given input data, as defined in App. B. Even after correction, all models except L-13B made this CONTEXT error (see Table 11d).

5.2.3 Insufficient Input
Correction: This is a data annotation problem because it is related to how the ToTTo dataset was created by pairing the sentence description and highlighting only a subset of cells from the Wikipedia table as Input (X). This problem is straightforward to fix by including the missing cells or headers from the tabular data.

In Table 5, when the choices for the Minnesota Amendment (i.e., ‘Referendum failed No’ and ‘Yes’ cells) were not included, all models incorrectly generated the favour of votes (‘approved’, ‘for’) and made an unsupported assumption regarding the turnout percentage. As shown in Table 5, all models corrected the errors after including the vote choices.

5.2.4 Longer Table Input
Correction: When dealing with straightforward tabular data, it was easier to correct the input either by i. correcting the cell values to ensure atomicity and/or ii. including the missing cells. However, this longer tabular input not only had over 100 rows but also included complex structures with nested column headers and row headers which posed difficulties in correcting the input.

Large models such as L-13B and T5-l could process tables with fewer than 20 rows for straightforward inputs. However, T5-b and L-7B struggled to produce factual information even for 20 rows. In our correction procedure, we chose an upper limit of 20 rows and 10 columns to simplify the longer tabular input. A simplified example of this problem with fewer rows is shown in Table 16 in App. F.

5.2.5 ToTTo: Politics Specific
Correction: We included appropriate semantic cues in the headers to help the model differentiate between the generic percentage of votes and the swing percentage of votes. For other errors, we expanded the abbreviation of coalition party names to avoid errors. In Table 10 from App. F, the original input only had ± in the header, resulting in errors from all models including L-13B. After explicitly including ‘± seats compared to the previous election’, both LLAMA 2 models corrected the previous error. However, both fine-tuned T5 models committed mistakes even after this correction.

Specific Semantic Generation Issue: In some cases, the model cannot determine whether a minister was ‘shortest-lived’ or ‘longest-lived’ only based on their lifespan, birth, and death years from the input. It may require additional context to produce accurate text. One such example is shown in Table 12. All models with the original data produced incorrect WORD error. After we included the ‘Total time in office’ cell and customized the prompt for LLAMA 2, both LLAMA 2 models corrected the WORD error. However, both T5 models did not show any improvement after including this detail. This could be due to the strong influence of the patterns learned from the fine-tuning process for similar tabular structures.

5.2.6 List of Leader Names
Correction: We did not change the order of the leader’s name in the title, but we addressed missing vote counts and leader names to map the relations for each record as shown in Table 13, Table 14, and Table 15 in the App. F.

In Table 14, the title leader ‘Chuck DeVore’ lost the election among four other party nominees. Both
T5 models made errors by focusing on ‘Chuck DeVore’ as the main candidate, resulting in WORDW and CONTEXTC errors. NUMBERU errors are present in all models due to missing votes. After including the votes and party details for all candidates, both LLAMA 2 models corrected all errors. However, both T5 models still made errors stating the title leader either defeated or being defeated by all candidates, possibly due to the learned patterns during fine-tuning on the ToTTo dataset.

In Table 15, the title leader, ‘Joseph Haslet’ ran as a candidate for Governor office in 1804 and 1807 but lost both elections. All models incorrectly generated Haslet won in both years. The correction is only limited to passing the party name and modifying the header from ‘Subject’ to ‘Candidate’. Despite this change, all models continued to generate errors. Both T5 models struggled to generate the right winning candidate in 5 out of 11 samples (Fig. 2a) for this input with the same set of headers.

5.3 T5 Models on Corrected Data
Based on the insights gathered from different input problems in Sec. 4, we followed the procedure and manually corrected the original tabular input for 210 samples (taken from 124 and 86 ‘errors’ category samples in Table 3). We ran both T5-b and T5-l models on the corrected data.

5.4 LLAMA 2 Models on Corrected Data
From the corrected data, as described in Sec. 5.3, we selected 40 challenging samples from the previous 210 samples as inputs to the LLAMA 2-7B (L-7B) and LLAMA 2-13B (L-13B) models. These samples were chosen to cover each of the input problem types described in Sec. 4, guaranteeing a thorough analysis. Within these 40 samples, 21 are associated with general input problems and the remaining 19 are specific to the politics input problems, as shown in Fig. 2c and Fig. 2d (in App. F).

We studied the zero-shot capabilities of L-7B and L-13B models for the 40 challenging samples. For each sample, we employed different prompts tailoring to the content of the tabular input (Kojima et al., 2022). Two common prompts we used are shown in App. C. For each sample and each of L-7B and L-13B, we examined the outputs before and after correcting the tabular input.

6 Results and Discussion
We manually annotated the outputs from all models working on corrected input data, following the same procedure outlined in Sec. 3.1 and defined in App. B. In this section, we summarize the results of the total input data corrections and discuss them. The previous section also described and discussed error analysis locally while presenting individual corrections.

6.1 Error Reductions in T5 Models
The input corrections significantly reduced factual errors, as evident in Table 6, which provides a comparison of error counts ‘before’ and ‘after’ input data fixes for each error category. It should be noted that no prompt or instruction was provided to these two fine-tuned T5 models.

<table>
<thead>
<tr>
<th>Category</th>
<th>T5-base (T5-b)</th>
<th>T5-large (T5-l)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Before</td>
<td>After</td>
</tr>
<tr>
<td>WORD</td>
<td>62</td>
<td>20</td>
</tr>
<tr>
<td>NAME</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>DATE_DIMENSION</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>NUMBER</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>OTHER</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>CONTEXT</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>ADDITION</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>NON-ENGLISH</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 6: Count of individual error annotations for 210 samples. The table compares the error counts between ‘before’ and ‘after’ applying input correction. Each sample can contain multiple errors.

6.2 Error Reductions in LLAMA 2 Models
In Table 7, we present the error analysis of 40 difficult samples, validated using L-7B and L-13B models with tailored prompts for each input. Outputs were manually error annotated, and the table compares the error counts before and after input correction. The L-7B and L-13B models showed reductions of 52% and 76% in errors, after addressing tabular input issues. The T5-b and T5-l models demonstrated error reductions of 46% and 53% respectively for the 40 difficult samples considered, which was shown for comparison purposes. The insights gathered from the individual error categories after input correction are mentioned below.

- WORDW errors, predominantly incorrect verbs and prepositions, were most common among all models. Post-correction, the L-13B model reduced this error category by 77%, while the T5-b, T5-l, and L-7B models
achieved reductions of 33%, 42%, and 34%, respectively.

• Input correction led to a reduction in NAME and CONTEXT errors across all models.

• The original input data exhibited more NUMBER errors in LLAMA 2 models compared to T5, which were significantly reduced after correction. Both LLAMA 2 models completely resolved DATE_DIMENSION and OTHER errors.

• ADDITION errors are more common in LLAMA 2 models than in T5. Despite including the prompt ‘Use only the information mentioned in the input table data’ and correcting the tabular data, both L-7B and L-13B models still produced five ADDITION errors each.

Table 8 shows two rows of data from our analysis of the 40 challenging samples. The first row shows that corrected data leads to an increased number of samples with ‘no error’ (Higher is better). However, the second row shows that omissions have increased after input corrections. We observed that the models omitted part of the information either when the corrected tabular data had multiple column variables for more than two records or when the tabular structure was complex. In the case of fine-tuned models, it learned to omit some information during the fine-tuning process. In our future work, we plan to study the reasons why this is happening.

6.3 Model-Specific Results for Different Input Problems

In App. E, Fig. 2 presents how the four models are performing before (left bars) and after input corrections (right bars) for each input problem type.

**Non-atomic cell values:** T5-l and L-13B models corrected over 90% of the errors for this problem type, single record and multiple records lacking atomicity (see red and green bars in Fig. 2, App. F). T5-b and L-7B models corrected over 60% of the errors but still struggled with a few samples even after correction. For example, Fig. 2b shows that T5-large was making more errors for the input problem type labelled ‘Multiple records lacking atomicity’ before correction (in red) and made significant reductions after correction (in green).

**Complex table:** Due to the limitations in some tabular inputs, which require additional context, T5-b could not fix the errors. T5-l omitted the error for one sample, while L-7B and L-13B models partially fixed this input problem (see Fig. 2).

**Insufficient input:** This data annotation problem fixed all factual errors in T5-b, T5-l and L-13B...
after correction except for L-7B which added additional information for one sample.

**Longer table input**: Large models such as L-13B, L-7B and T5-I corrected factual errors for straightforward inputs, especially for tables with fewer than 20 rows. However, T5-b struggled the most to produce factual information.

**Politics specific semantic issue**: For the corrected input, L-13B fixed the factual mistakes for 5 out of 8 samples and L-7B fixed factual errors for 4 out of 8 samples. Both T5 models corrected 3 out of 8 samples (see Fig. 2).

**List of leader names**: T5 models struggled the most to correct factual errors for this problem. After correction, the L-7B model corrected three samples but produced factual errors for the remaining eight samples. L-13B made factual errors only for two samples (see Fig. 2).

While some models struggle with specific inputs, particularly regarding leader name order, tables requiring additional context for complex tabular structures, and semantic issues with symbols, our overall results indicate that correcting tabular inputs improves model outputs.

### 7 Inter-Annotation Experiment

One of the authors manually annotated errors in 1,508 outputs before input correction and 169 problematic outputs after correction (a total of 1,677 from both T5 models). Similarly, the annotation for both LLAMA 2 models covered 160 outputs before and after input correction.

Two additional annotators annotated 60 outputs each, generated by four different models both before and after input correction. We provided them with a detailed document that included definitions of error categories, guidelines, tabular inputs and output texts for error annotation. Annotators followed the guidelines and marked the errors. Each annotator spent approximately 3 hours to complete this experiment.

The annotated errors are shown in a confusion matrix in Table 9, where the ‘all agree’ column signifies unanimous agreement on error types, while other columns show unique error selections.

<table>
<thead>
<tr>
<th>ERROR</th>
<th>ALL AGREE</th>
<th>WORD</th>
<th>NAME</th>
<th>DATE_DIM</th>
<th>NUMBER</th>
<th>OTHER</th>
<th>CONTEXT</th>
<th>ADDITION</th>
<th>NO ERROR</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>9</td>
<td>3</td>
<td>0</td>
<td>31</td>
</tr>
<tr>
<td>NAME</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>DATE_DIM</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>NUMBER</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>OTHER</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>CONTEXT</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>26</td>
</tr>
<tr>
<td>ADDITION</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>NO ERROR</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>16</td>
</tr>
</tbody>
</table>

**Table 9**: Fleiss’ Kappa coefficient: overall agreement on 60 samples among three annotators. ‘All agree’ column signifies unanimous agreement on error types, while other columns show unique error selections.

Fleiss’ kappa overall agreement for three annotators, \( k = (pa - pe)/(1 - pe) = 0.622 \)

8 Conclusion

This paper presented a study that quantitatively demonstrates that fixing input problems such as insufficient data and data records containing non-atomic content improves the factual accuracy of output text by as high as 76% for one of the study models. Correcting inputs also seems to improve the number of entirely error-free output texts. However, we still need to investigate why errors categorised as ‘omissions’ increase after input corrections. In our future work, we aim to explore other tabular datasets for problems with input data and study the generalization of the fixes explored in the current work.

We release our error annotations from our human evaluation on [https://github.com/BarkaviSJ/totto_politics_human_annotations](https://github.com/BarkaviSJ/totto_politics_human_annotations).
Limitations

We acknowledge some limitations in this work. First, we only looked at ToTTo and our scope of corrections to tabular input is limited to errors identified within the politics domain of the ToTTo validation set. Second, we did not extend the correction of tabular input for the politics domain to the training split of the ToTTo dataset due to the time-consuming process of handling different table headers and metadata. Third, our experiment results are restricted to two specific models (T5 and LLAMA 2) and may not generalize to other models.

In our future work, we aim to simplify the definitions of the ‘context’ and ‘addition’ error categories, as the annotation experiment revealed disagreement in choosing these error types for some samples, despite annotators marking the same error token.

Ethics Statement

This work seeks to address input problems in non-standard tabular structures to reduce factual errors in the output text. We utilized the open-source dataset, ToTTo and maintained the same ground-truth generation as the original dataset. Our input correction did not introduce any further social bias to this dataset. We adopted an error annotation methodology to annotate factual errors and one of the authors performed manual error analysis for this complete study. We sought consent from two additional annotators, the annotators volunteered to participate and annotated errors for 60 output texts each. They had the right to withdraw from the study at any point without facing any consequences. We provided necessary guidelines, instructions and examples for them to annotate errors.

Acknowledgements

We thank Craig Thomson and Adarsa Sivaprasad for their hard work in helping with the annotations in this paper. We thank the anonymous reviewers for their detailed feedback and suggestions which have significantly improved this work. We also thank the NLG (CLAN) reading group at the University of Aberdeen for their invaluable feedback.

References


Barkavi Sundararajan, Somayajulu Sripada, and Ehud Reiter. 2022. Error analysis of ToTTo table-to-text


Appendices

A  Model fine-tuning specifications

The first model, T5-base (T5-b), was fine-tuned on the full ToTTo training set of 120,761 samples on a commodity server with a GeForce RTX 2080 GPU. The training took around seven days. The second model, T5-large (T5-l), was fine-tuned on a subset of 50,000 ToTTo samples on a secure cloud instance with an NVIDIA A100 GPU, completing in around 48 hours. Both models were fine-tuned using a constant learning rate of 0.0001, with the encoder’s maximum length set to 512 tokens and the decoder’s maximum length set to 128 tokens for ToTTo’s generation task (Kale and Rastogi, 2020). Single-precision floating-point format (FP32) was employed for training on their respective GPU servers. The batch size, beam size and training steps for each model are shown in Fig. 1.

<table>
<thead>
<tr>
<th>Models</th>
<th>Batch size</th>
<th>Beam size</th>
<th>Training steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>T5-base</td>
<td>2</td>
<td>10</td>
<td>180,000</td>
</tr>
<tr>
<td>T5-large</td>
<td>4</td>
<td>5</td>
<td>9,000</td>
</tr>
</tbody>
</table>

Figure 1:  Model Specifications

B  Inter-Annotation Procedure for Participants

B.1 Overview

The Input Data from the ToTTo dataset, includes the Page title, Section Title and a Table with highlighted cells in yellow. These key parameters are conditionally used for training the neural models to summarize a meaningful and factual Text (as Output) focusing on: (i). highlighted cells in the table (ii). their corresponding header values (iii) The main Title and (iv) The Section Title.

For each of these tabular input data, we provided outputs generated by different neural language models to annotators. Our goal is to evaluate whether the neural outputs remain faithful and produce factually accurate information based on the four parameters from the tabular input. The complete table, including the non-highlighted cells, is provided to offer a clearer understanding of the error annotation task.

B.2 Domain

The inputs provided are specific to the domain of politics, sourced from Wikipedia tables (as part of the open-domain ToTTo dataset). The political data within these tables is not limited to a single demography. Instead, it encompasses various details from the election processes across multiple countries, including:

- Election specifics such as Presidential, state, by-elections, council, district, Legislative Assembly, and other elections unique to particular countries.
- Information about Governors, Mayors, Ministers, and Ambassadors (about Foreign Affairs).
- Details regarding the Speaker of the Assembly.

The first item related to election details is predominantly used in this error annotation task.

B.3 Error Annotation guidelines

We are only interested if the highlighted cell values from the table were used to produce a factually correct sentence. Please also pay attention to the non-highlighted cells in the same row as the highlighted cells, as this might be required in some inputs to generate a meaningful sentence. Other non-highlighted values in the table are not expected to be used for your evaluation. Please read through the output texts and annotate cases for the error categories as mentioned below. Each error is denoted with a superscript for better readability.

NAME

- When names of the Party, Leader, place (Electorate), Ambassador etc., are wrong (mostly nouns).
- Annotation includes both single tokens or multiple tokens to include the complete names.

Example Output text: Urban Ahlin\textsuperscript{N} is the Deputy Speaker of the Riksdag. Remarks: NAME error because the correct deputy speaker was Tobias Billström as per the tabular data.

- Example Output text: Kansas was won by Mitt Romney, Paul Ryan, Barack Obama\textsuperscript{N}, and Joe Biden\textsuperscript{N}, with Romney winning...
59.66% of the popular vote, six electoral votes and 38.05 percent. In this example, Barack Obama and Joe Biden are two NAME errors because they did not win the election.

- In general, Wednesday instead of Tuesday is a NAME error but May instead of April is a DATE DIMENSION error.

**NUMBER**

- When the number of seats and/or the number of votes and/or % of votes are incorrect. A single token is marked as an error.

- When the A-party won with a majority of 5.5%. But the correct one is 4.4%. 5.5% is a NUMBER error.

- Output: The voter turnout was 8.90%, with 10,052 votes. Remarks: The actual turnout was 81.90%. Please note: the error here is NOT with the comma used as decimal (as it is an acceptable decimal operator for international use); Error because the number 81.90 was incorrect.

**DATE DIMENSION**

- When the Date and/or Month and/or Year are wrong in the generated text, it is annotated as one error.

- Example Output text: Cletus Avoka was the Minister for the Interior in the Mills government from 2009 to 2012. Remarks: 2010 is the right end term of the year.

- As a general note, if the Output text did not capture Month and/or Date, but has the correct year, then this is NOT an ERROR. It could go to OMISSION with remarks, Omission of Date and Month.

**WORD**

- When incorrect words such as verbs, prepositions, adjectives, adverbs and conjunctions are found in the output.

- Single token is marked as a WORD error in most cases. Multiple tokens are annotated when the auxiliary verb (was), an extension of a prepositional phrase (along with) and others are incorrect.

- Example Output text: Carly Fiorina defeated Republican Tom Campbell with 56.4% of the vote to DeVore’s 19.3%, along with Al Ramirez and Tim Kalemkarian. Remarks: Fiorina independently defeated all the leaders, so ‘along with’ is wrong.

- Example Output text: Ling won the 2016 senate district against Democrat Josh Newman, with 49.6 percent of the vote. Remarks: Ling lost the election as per the tabular data.

- Some of the common WORD errors found in this data are won, defeated, lost, succeeded, adjectives such as current governor other prepositions (since, in and so on).

**CONTEXT**

- When the model’s output presents information that contradicts or makes unsupported assumptions about the given input data. Group of tokens/span of text are annotated as CONTEXT error.

- It can sometimes be tricky to check for this type of error. Please follow the below sequence before marking this error.

  - In case of simple misrepresentation based on the information in the input data, it would be easier to mark the token as NAME, NUMBER, DATE DIMENSION or WORD error.
  
  - In the case of a complex table structure, the outputs are likely to mess up completely with the overall information in the provided input data. In this case, it is hard to mark individual errors. Please go ahead and mark the group of tokens/span of text and annotate it as a CONTEXT error.

  - For example, the output is: In the 2006 election for mayor of Florence Pendleton, Michael D. Brown received 62,415 votes while Philip Pannell received 21,552 votes and write-in candidates received 1,363 votes. Annotation remarks:

    - for mayor of Florence Pendleton - the name of a person is misrepresented as electorate (jurisdiction) in the output.
write-in candidates\textsuperscript{C} - this implies there was more than one write-in candidate.

**ADDITION\textsuperscript{A}**

- When the model’s outputs have *added words, phrases, or details* that either diverge from the input’s main topic or are unsupported by the given context.
- Single or group of tokens are annotated as ADDITION error.
- Example Output text: 2007 Algerian legislative election was held on May 17, 2007\textsuperscript{A}. The results were as follows: 24 political parties won a total of 389 seats in the National Assembly. **Remarks:** The date and month are additional information, that are not provided in the input. This is marked as an ADDITION error. Other details in the output are correct.

**OTHER\textsuperscript{O}**

- When the output repeats the same input multiple times producing garbage data. In some cases, when the table data has the political party name in the abbreviation, it produces garbage output. For example, when the tabular data has a party name in abbreviation, it tries to produce a strange output. Output text: GSSDULSVDHSS\textsuperscript{O} gained 5.31\% of the vote
- When the output is incomplete for longer table input or when the output repeats the same input multiple times without producing a complete text.
- When punctuation symbols are placed in inappropriate places, for example, an apostrophe is missed for the Name of the Leader or Place.

**NON-ENGLISH\textsuperscript{NE}**

- when the Unicode characters in non-English names are either replaced with special characters or when these Unicode characters are omitted.
- For example, Pawe Gra\textsuperscript{NE} is a member of Sejm. **Remarks:** Paweł Graś is the correct name here.

**C LLAMA 2 prompts**

The below prompt is for the corrected tabular data when the input table has party name, candidate and votes.

‘Given the input table data, the task is to:
(i). Identify the party name, candidate name, and the number of votes received by each candidate.
(ii). Determine the winner based on the highest number of votes. Then, put together the gathered information from (i) and (ii) into a single coherent sentence. Input table data: <Linearized table data>’

Below is one of the prompts used for the original table data without mentioning any specific fields.

‘The task is to summarize the information from the given input table data into a single coherent sentence. Use only the information mentioned in the input table data. Input table data is: <Linearized table data>’

**D Steps for Correcting ToTTo Tabular Input**

In addition to the correction procedure detailed in Sec. 5.2, which provides examples of the corrections applied to each tabular input problems, we present a supplementary pseudocode in this section. In algorithm 1, we have a generic function that takes tabular data and title (metadata) from ToTTo as input parameters. This main algorithm executes six different functions.

The first two functions, DATA\textsupersample{}FOR\textsupersample{}LONG\textsupersample{}TABLE and IDENTIFY\textsupersample{}LEADER\textsupersample{}NAME\textsupersample{}ORDER provide insights on the input problems leading to factual errors in the outputs but do not correct the tabular input. The other four functions, namely CORRECT\textsupersample{}NON\textsupersample{}ATOMIC\textsupersample{}CELLS, UPDATE\textsupersample{}HEADERS, ADDRESS\textsupersample{}MISSING\textsupersample{}VALUES, and REPLACE\textsupersample{}SYMBOLS correct the input problems as presented in algorithm 4, algorithm 5, algorithm 6, and algorithm 7. We provide brief descriptions for each algorithm in the caption and comments.

**E Input Problem Types for four models**

Fig. 2 shows how each of the four models is performing before and after data corrections for each of the problem types. The left bars for each input
Algorithm 1 Manual Correction Procedure for ToTTo Tabular Input (elaborated in Sec. 5.2). This main function takes tabular data and title details as input parameters and returns the corrected tabular data. In all functions, we excluded row and column indices for simplicity and readability.

function MAINCORRECTIONPROCEDURE(tabularData, title)
    if not DATASIZEMANAGEABLEFORTABLE(tabularData) then
        return "Please simplify the tabular data with fewer records", null
    end if
    leaderName, recordedData ← IDENTIFYLEADERNAMEORDER(tabularData, title)
    correctionsMade ← false
    correctionsMade ← CORRECTNONATOMICCELLS(tabularData) or correctionsMade
    correctionsMade ← UPDATEHEADERS(tabularData) or correctionsMade
    correctionsMade ← ADDRESSMISSINGVALUES(tabularData) or correctionsMade
    if correctionsMade then
        return (tabularData, "Leader Data: " + recordedData) ▷ Returns corrected input and leader data
    else
        return (tabularData, "Leader Data: " + recordedData) ▷ Returns original input (if no issues) and leader data
    end if
end function

Algorithm 2 Verify the number of rows and columns in Longer Tabular Input (discussed in Sec. 5.2.4). This function validates the maximum allowable number of rows and columns for the tabular data and returns true or false to the main function.

function DATASIZEMANAGEABLEFORTABLE(tabularData)
    maxRows ← 20
    maxCols ← 10
    return (length(tabularData) ≤ maxRows) and (length(tabularData[0]) ≤ maxCols)
end function
Algorithm 3 Identify Leader Name Order in Title and Table Rows (discussed in Sec. 5.2.6). This function verifies three main scenarios for the order of leader names in the input. It returns a tuple with title information for leader name and a list of leader names from tabular input depending on the scenario. Description for each scenario is briefly commented.

```plaintext
function IDENTIFYLEADERNAMEORDER(tabularData, title)
    leaderNameFromTitle ← ExtractLeaderNameFromTitle(title)
    leaderNamesFromRows ← []
    for each row in tabularData do
        for each cell in current row do
            if leader_name is found in cell then
                Add leader_name to leaderNamesFromRows ▷ Record leader’s names from rows
            end if
        end for
    end for
    ▷ Scenario 1: Leader’s name found in title
    if leaderNameFromTitle is not None then
        recordedData ← [] ▷ To store sequential order of leader names
        Add leaderNameFromTitle to recordedData
        for each leader_name in leaderNamesFromRows do
            Add leader_name to recordedData
        end for
        return (leaderNameFromTitle, recordedData) ▷ for e.g., this could return ("b", ["a", "b", "c"]) end if
    else ▷ Scenario 2: Leader’s name not found in title
    if length of leaderNamesFromRows > 0 then ▷ if leader’s name is in at least one row
        recordedData ← "Leader name not in title"
        for each leader_name in leaderNamesFromRows do
            Add leader_name to recordedData
        end for
        return (leaderNameFromTitle, recordedData) ▷ for e.g., this could return (None, ["Leader name not in title", "a", "b", "c"])
    else ▷ Scenario 3: Leader name neither in title nor in rows
        return (leaderNameFromTitle, "Leader name not in table") ▷ for e.g., this could return (None, ["Leader name not in table"]) end if
end function
```
Algorithm 4 Correct Non-Atomic Cells (discussed in Sec. 5.2.1). If a non-atomic cell is found, the CorrectCell function separates multiple atomic values into individual columns. We follow our manual correction procedure in CellIsNonAtomic and CorrectCell. The function returns the corrected tabular data along with a flag indicating if any corrections were made.

```plaintext
function CorrectNonAtomicCells(tabularData)
    correctionsMade ← false
    for all row in tabularData do
        for all cell in row do
            if CellIsNonAtomic(cell) then
                correctedCell ← CorrectCell(cell) ▷ Separate multiple values into individual columns
                tabularData[cell] ← correctedCell ▷ Update the corrected cell value
                correctionsMade ← true
            end if
        end for
    end for
    return (tabularData, correctionsMade)
end function
```

Algorithm 5 Update Column and Row Headers to Atomic cells (discussed in Sec. 5.2.4). For each cell, the tabular input has header_value as true or false. When the header value is true, we manually verify the function HeaderIsIncorrect and correct the logic in UpdateHeader(cell). The function returns the corrected tabular data with corrections made flag.

```plaintext
function CorrectHeaders(tabularData)
    correctionsMade ← false
    for all row in tabularData do
        for all cell in row do
            if cell.header_value = true then
                if HeaderIsIncorrect(cell) then
                    correctedHeader ← UpdateHeader(cell) ▷ Update column and row headers
                    tabularData[cell] ← correctedHeader ▷ Update the corrected header
                    correctionsMade ← true
                end if
            end if
        end for
    end for
    return (tabularData, correctionsMade)
end function
```
Algorithm 6 Address Missing cell Values (discussed in Sec. 5.2.3). For each row, we verify the missing cell values in RowHasMissingValues and pass the right cell values in FillCellValues through a manual process. The function returns the corrected tabular data with corrections made flag.

```
function ADDRESSMISSINGVALUES(tabularData)
    correctionsMade ← false
    for all row in tabularData do
        if RowHasMissingValues(row) then
            correctedRow ← FillCellValues(row) ▶ Pass missing cells in the row
            tabularData[row] ← correctedRow
            correctionsMade ← true
        end if
    end for
    return (tabularData, correctionsMade)
end function
```

Algorithm 7 Replace Politics Domain-Specific Symbols and Abbreviate Party Names with Correct Semantics/words (discussed in Sec. 5.2.5). For all cells in the table, we verify containsDomainSpecificSymbols and containsPartyAbbreviations functions and correct the values using substituteSymbolWithEquivalent and containsPartyAbbreviations through a manual process. The function returns the corrected tabular data with corrections made flag.

```
function REPLACESYMBOLS(tabularData)
    correctionsMade ← false
    for all cell in tabularData do
        if containsDomainSpecificSymbols(cell) then
            correctedValue ← substituteSymbolWithEquivalent(cell) ▶ Replace symbols with words
            tabularData[cell] ← correctedValue ▶ Update the corrected cell value
            correctionsMade ← true
        else if containsPartyAbbreviations(cell) then
            correctedValue ← abbreviatePartyNames(cell) ▶ Abbreviate party names
            tabularData[cell] ← correctedValue ▶ Update the corrected cell value
            correctionsMade ← true
        end if
    end for
    return (tabularData, correctionsMade)
end function
```
problem type represent output scores before data correction, while the right bars represent scores after data correction. Colour coding of output scores helps to identify and understand errors: green indicates no error, red indicates an error, and yellow indicates an omission.

The number of samples for each model differs because we focus on outputs only when the model had ‘errors’ or ‘omissions’ before correction. This approach emphasizes the actual improvement in corrections. For example, T5-large had errors in 37 samples before correction (left bar), the after-correction (right bar) also shows improvements for the same 37 samples.

F Input Data Corrections and Output Text

In this section, the input tabular data with outputs supplements the examples elaborated in Sec. 5.2 for each of the input problems as presented from Table 10 to Table 16.
In the 2012 Cardiff South and Penarth by-election, the Liberal Democrats lost by 11.5% of the vote compared to the previous election.

Table 10: ToTTo specific: Swing percentage difference with previous election, as detailed in Sec. 5.2.5. Before fix represents the output generated from input Table 10a and after fix represents the output generated after explicitly changing the header, as shown in input Table 10b.
Figure 2: Comparison of different input problem types before (original data) and after data correction across four models. It shows the count of unique samples for three categories: (i) errors, (ii) omissions, and (iii) no errors.
Page Title: 1914 United States Senate elections
Section Title: Races leading to the 64th Congress

<table>
<thead>
<tr>
<th>State</th>
<th>Incumbent</th>
<th>Party</th>
<th>Results</th>
<th>Candidates</th>
</tr>
</thead>
<tbody>
<tr>
<td>South Dakota</td>
<td>Coe I. Crawford</td>
<td>Republican</td>
<td>Incumbent lost renomination. New senator elected. Democratic gain.</td>
<td>Edwin S. Johnson (Democratic) 48.32% Charles H. Burke (Republican) 44.47%</td>
</tr>
</tbody>
</table>

(b) Outputs with Original Input before Input Data Correction

**T5-base:** In South Dakota, Republican Coe I. Crawford lost renomination to Democrat Edwin S. Johnson.  
**T5-large:** South Dakota: Republican incumbent Coe I. Crawford lost his bid for the seat to Democratic challenger Edwin S. Johnson.

**LLAMA 2-7B:** 1914 United States Senate elections in South Dakota were held on November 3\(^{\text{rd}}\), 1914. Incumbent Republican Senator Coe I. Crawford lost renomination to Democratic candidate Edwin S. Johnson\(^{\text{c}}\). Johnson defeated Charles H. Dunnigan\(^{\text{c}}\) by a margin of \(48.32\%\) to \(47.56\%\). This was a Democratic gain from the previous election.

**LLAMA 2-13B:** 1914 United States Senate elections were held on November 3\(^{\text{rd}}\), 1914 to elect members of the United States Senate in various states. The Republican Party lost one seat and the Democratic Party gained one seat.

(c) Input Table with Corrected Cells highlighted in Yellow

Page Title: 1914 United States Senate elections
Section Title: Races leading to the 64th Congress

<table>
<thead>
<tr>
<th>State</th>
<th>Incumbent Senator</th>
<th>Incumbent Party</th>
<th>Results</th>
<th>Candidate Senator</th>
<th>Candidate Party</th>
<th>Candidate votes %</th>
</tr>
</thead>
<tbody>
<tr>
<td>South Dakota</td>
<td>Coe I. Crawford</td>
<td>Republican</td>
<td>Incumbent lost renomination. New senator elected. Democratic gain.</td>
<td>Edwin S. Johnson</td>
<td>Democratic</td>
<td>48.32%</td>
</tr>
<tr>
<td>South Dakota</td>
<td>Coe I. Crawford</td>
<td>Republican</td>
<td>Incumbent lost renomination. New senator elected. Democratic gain.</td>
<td>Charles H. Burke</td>
<td>Republican</td>
<td>44.47%</td>
</tr>
</tbody>
</table>

(d) Outputs After Input Data Correction

**T5-base:** In South Dakota, Republican Coe I. Crawford lost renomination to Democrat Edwin S. Johnson.  
**T5-large:** South Dakota: Republican incumbent Coe I. Crawford lost his bid for the seat to Democratic challenger Edwin S. Johnson.

**LLAMA 2-7B:** 1914 United States Senate elections were held on November 3\(^{\text{rd}}\), 1914. In South Dakota, Coe I. Crawford, the Republican incumbent Senator lost renomination to Edwin S. Johnson, a Democratic candidate. Charles H., the Republican candidate also ran for the seat but he was defeated by Johnson. Therefore, it is a Democratic gain in this election.

**LLAMA 2-13B:** 1914 United States Senate elections were held on November 3\(^{\text{rd}}\), 1914 to elect members of the United States Senate in various states. Races leading to the 64th Congress included South Dakota where Coe I. Crawford (Republican) lost renomination and Edwin S. Johnson (Democratic) was elected.

Table 11: Complex Table input problems, as detailed in Sec. 5.2.2. Difficult to fix this complex input type, yet LLAMA 2-13B produced most of the facts right except for ADDITION error. Lost renomination ‘to Democrat candidate’ is a CONTEXT error, because the model made unsupported assumptions about the given input. Losing renomination is when the current senator failed to be nominated in the primary to stand for reelection.
Page Title: List of Japanese prime ministers by longevity
Section Title: Longevity of Japanese prime ministers

<table>
<thead>
<tr>
<th>Prime Minister</th>
<th>Total time in Office</th>
<th>Date of Death</th>
<th>Lifespan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sanetomi Sanjō</td>
<td>60 days</td>
<td>28 Feb 1891</td>
<td>53 years, 352 days</td>
</tr>
</tbody>
</table>

(c) Outputs Before and After Input Data Correction

T5-base and T5-large before and after fix (no improvement): Sanjō Sanetomi was the longest-lived prime minister in Japan at 53 years, 352 days.

LLAMA 2-7B before fix: The longest-serving Japanese prime minister was Sanjō Sanetomi who served for 53 years and 352 days.

LLAMA 2-7B after data fix: Sanjō Sanetomi was the Japanese prime minister and served for a total of 60 days in office. He died at age 53 years and 352 days.

LLAMA 2-13B before data fix: The longest-lived Japanese prime minister was Sanjō Sanetomi, who served for 53 years and 352 days.

LLAMA 2-13B after data fix: Sanjō Sanetomi was the prime minister of Japan for a total of 60 days. He lived for 53 years and 352 days.

Table 12: ToTTo specific - WORD error made by all models before fix, as detailed in Sec. 5.2.5. After including ‘Total time in Office’, LLAMA 2 models fixed the ‘longest-lived’ word error and introduced other addition errors. We then revised the prompt and fixed the addition errors. T5 models had no improvements after including information.

Table 13: Fixing ToTTo specific input problems: List of leader names for two leaders. ‘Before’ shows WORD errors due to missing vote percentages and party names for a leader. ‘After’ shows these errors are resolved by all models when the missing values are provided. Key point is T5 model did not produce error even though the title leader ‘Ling Ling Chang’ lost the election.
Table 14: ToTTo specific input problems: List of Leader names for five leaders, as detailed in Sec. 5.2.6. All models swapped the defeated leader names and/or % of votes, resulting in WORD, NUMBER and CONTEXT errors. After including the party names and % of votes for all leaders, LLAMA 2 models corrected the errors. Fine-tuned T5 models still made errors. This could be because of the pattern learned from the fine-tuned training data of ToTTo.
(a) Input Table with Original Cells highlighted in Yellow
Page Title: Joseph Haslet
Section Title: Almanac

<table>
<thead>
<tr>
<th>Year</th>
<th>Office</th>
<th>Subject</th>
<th>Party</th>
<th>% Votes</th>
<th>Opponent</th>
<th>Party</th>
<th>% Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1804</td>
<td>Governor</td>
<td>Joseph Haslet</td>
<td>Republican</td>
<td>48%</td>
<td>Nathaniel Mitchell</td>
<td>Federalist</td>
<td>52%</td>
</tr>
<tr>
<td>1807</td>
<td>Governor</td>
<td>Joseph Haslet</td>
<td>Republican</td>
<td>48%</td>
<td>George Truitt</td>
<td>Federalist</td>
<td>52%</td>
</tr>
</tbody>
</table>

(b) Input Table with Corrected Cells highlighted in Yellow
Page Title: Joseph Haslet
Section Title: Almanac

<table>
<thead>
<tr>
<th>Year</th>
<th>Office</th>
<th>Candidate</th>
<th>Party</th>
<th>% Votes</th>
<th>Opponent</th>
<th>Party</th>
<th>% Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1804</td>
<td>Governor</td>
<td>Joseph Haslet</td>
<td>Republican</td>
<td>48%</td>
<td>Nathaniel Mitchell</td>
<td>Federalist</td>
<td>52%</td>
</tr>
<tr>
<td>1807</td>
<td>Governor</td>
<td>Joseph Haslet</td>
<td>Republican</td>
<td>48%</td>
<td>George Truitt</td>
<td>Federalist</td>
<td>52%</td>
</tr>
</tbody>
</table>

(c) Outputs Before and After Input Data Correction

T5-base before fix: Joseph Haslet was elected Governor in 1804, defeating Federalist Party candidate Nathaniel Mitchell in the election of 1807.

T5-base after fix: Joseph Haslet was elected Governor of the Republican Party in 1804 and 1807, defeating the Federalist Party’s Nathaniel Mitchell and George Truitt.

T5-large before fix: Joseph Haslet was elected Governor in 1804, defeating Federalist Party candidate Nathaniel Mitchell in the election of 1807.

T5-large after fix: In 1804 and 1807, Republican Joseph Haslet defeated Federalist Nathaniel Mitchell and Federalist George Truitt.

LLAMA 2-13B before fix: 1804, Joseph Haslet was elected governor with 48% of the vote. In 1807, he was re-elected with 48% of the vote.

LLAMA 2-13B after fix: 1804: Joseph Haslet (Republican) won with 48% of the votes. 1807: George Truitt (Federalist) won with 52% of the votes.

Table 15: ToTTo specific input problems: List of Leader names, as detailed in Sec. 5.2.6. Correction for this input is including the party name and modifying the header ‘Subject’ to ‘Candidate’. All models generated Joseph Haslet to be the winning candidate including the LLAMA 2 models that had specific instructions.
(a) Input Table with Original Cells highlighted in Yellow

<table>
<thead>
<tr>
<th>Rank</th>
<th>Representative</th>
<th>Party</th>
<th>District</th>
<th>Seniority date</th>
<th>Term details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joseph G. Cannon</td>
<td>R</td>
<td>IL-18</td>
<td>March 4, 1915</td>
<td>Twenty-three non-consecutive terms</td>
</tr>
<tr>
<td>2</td>
<td>Frederick H. Gillett</td>
<td>R</td>
<td>MA-2</td>
<td>March 4, 1893</td>
<td>Fifteen consecutive terms</td>
</tr>
<tr>
<td>3</td>
<td>Henry A. Cooper</td>
<td>R</td>
<td>WI-1</td>
<td>March 4, 1921</td>
<td>Fourteen non-consecutive terms</td>
</tr>
<tr>
<td>4</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>One term</td>
</tr>
<tr>
<td>5</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>One term</td>
</tr>
<tr>
<td>433</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>One term</td>
</tr>
<tr>
<td>434</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>One term</td>
</tr>
<tr>
<td>435</td>
<td>Adam M. Wyant</td>
<td>PA-22</td>
<td>WI-1</td>
<td>March 4, 1921</td>
<td>One term</td>
</tr>
</tbody>
</table>

Members joining the House, after the start of the Congress

... ... ... ... ...

(b) Outputs with Original Input before Input Data Correction

T5-large before fix: The 435th member of the United States House of Representatives in the 65th Congress was a member who joined the House after the start of Congress.

LLAMA2-13B before fix: List of Representatives by seniority The table below lists the 100.

(c) Input Table with Corrected Cells highlighted in Yellow

<table>
<thead>
<tr>
<th>Rank</th>
<th>Representative</th>
<th>Party</th>
<th>District</th>
<th>Seniority date</th>
<th>Term details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joseph G. Cannon</td>
<td>R</td>
<td>IL-18</td>
<td>March 4, 1915</td>
<td>Twenty-three non-consecutive terms</td>
</tr>
<tr>
<td>2</td>
<td>Frederick H. Gillett</td>
<td>R</td>
<td>MA-2</td>
<td>March 4, 1893</td>
<td>Fifteen consecutive terms</td>
</tr>
<tr>
<td>3</td>
<td>Henry A. Cooper</td>
<td>R</td>
<td>WI-1</td>
<td>March 4, 1921</td>
<td>Fourteen non-consecutive terms</td>
</tr>
<tr>
<td>4</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>One term</td>
</tr>
<tr>
<td>5</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>One term</td>
</tr>
<tr>
<td>433</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>One term</td>
</tr>
<tr>
<td>434</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>One term</td>
</tr>
<tr>
<td>435</td>
<td>Adam M. Wyant</td>
<td>PA-22</td>
<td>WI-1</td>
<td>March 4, 1921</td>
<td>One term</td>
</tr>
</tbody>
</table>

(d) Outputs After Input Data Correction

T5-large after fix: There are 435 members of the United States House of Representatives in the 65th Congress.

LLAMA2-13B after fix: The table shows that there are 435 representatives in the 65th Congress, and they are ranked according to their seniority.

Table 16: Longer Table input problems, as detailed in Sec. 5.2.4.

(a). Original tabular input had 435 rows with term details and other details from nested headers (irrelevant details).
(b). The fine-tuned T5-large model hallucinated the output text with the irrelevant header information (nested header issue). LLAMA 2-13B also struggled to produce the right text and generated incomplete output. LLAMA 2-7B produced longer garbage output.
(c). First, we fixed the nested header by creating a separate column. Then, we only passed 20 records (first 10 rank and 426 to 435 rank) along with title information. We did not pass the irrelevant term details to the corrected data.
(d). This simplified input records with relevant header details (Rank) fixed the errors in both models.