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[1] 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 (Sellam 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:

[1] https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Accessibility/Data_tables_tutorial#Column_headers:_bad_example

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 7347–7373

(a) Original cells highlighted in Yellow as Input

1996 United States House of Representatives election

Party	Candidate	Votes	%
Democratic	Eleanor Holmes Norton (inc.)	134,996	90.00
Republican	Sprague Simonds	11,306	7.54

LLAMA 2-13B Output for tabular input a:
Eleanor Holmes Norton (inc.) won with **7.54%**^{U} of the vote. Sprague Simonds was the Republican candidate and received **22.38%**^{U} of the vote.

(b) Corrected Tabular data by including relevant cells

1996 United States House of Representatives election

Party	Candidate	Votes	%
Democratic	Eleanor Holmes Norton (inc.)	134,996	90.00
Republican	Sprague Simonds	11,306	7.54

LLAMA 2-13B Output for tabular input b:
Democratic Party candidate Eleanor Holmes Norton won with 90% of the vote. Republican Party candidate Sprague Simonds received 7.54%.

Table 1: ToTTo example: Highlighted cells in yellow are passed as input to the model. Passing the appropriate cells (i.e., % votes and party name) as input, as shown in Table 1b fixes the factual errors. Compare the Table 1b output (with no errors) to the Table 1a output (with NUMBER errors denoted by a superscript U).

<page_title> 1996 United States House of Representatives election </page_title> <table> <cell> Eleanor Holmes Norton (inc.) <col_header> Candidate </col_header> </cell> <cell> Republican <col_header> Party </col_header> </cell> <cell> Sprague Simonds <col_header> Candidate </col_header> </cell> <cell> 7.54 <col_header> % </col_header> </cell> </table>

Table 2: Linearized Input from (a) and (b) is passed as input to LLAMA 2-13B model. For example, we present the Linearized Input for Table 1a here. The corresponding input for Table 1b is not shown, but it will include all related cells highlighted in yellow in (b).

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[2]. The inter-annotation agreement on the error annotation was good, with a Fleiss' Kappa of 0.622 (Sec. 7).

[2]Error annotation guidelines and sample annotations from our human evaluation is available at https://github.com/BarkaviSJ/totto_politics_human_annotations

1.1 Table to Text dataset, ToTTo

ToTTo is an open-domain English language dataset (Parikh et al., 2020), where the input X is taken from Wikipedia table T, which includes the table's metadata (such as title) and a set of highlighted cells, along with their headers. This structured information is flattened to create a linearized text representation of the table, as mentioned in Table 2. This crowdsourced dataset is paired with a natural language description, denoted as output Y, comprising a sequence of tokens $y_1, y_2, \ldots, y_n$, which provides a coherent summary of the content present in the input table X.

These input-output pairs from the ToTTo dataset can be used for fine-tuning or prompting the neural language models. As shown in Table 1a, the Input X is often observed to be problematic and fixing these problems is the main focus of this paper.

2 Related Work

Prior work on ToTTo: Wang et al. (2022) proposed a framework, LATTICE, that preserves the structural information of the cell values (tokens) within the same rows or columns, and by removing the attention mechanism flow for other unrelated tokens. Hu et al. (2023) incorporates content planning and generation from Su et al. (2021) and synthetically added noisy cells in their fine-tuning regime. While these approaches are model agnostic and improved automatic metric scores such as

BLEU (Papineni et al., 2002), PARENT (Dhingra et al., 2019) and BLEURT (Sellam et al., 2020) by few points in the leaderboard [3], 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 WORD^{W}, NAME^{N}, DATE_DIMENSION^{D}, NUMBER^{U}, CONTEXT^{C} and OTHER^{O}) from our error categories. Extrinsic refers to our ADDITION^{A} 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^{W}, NAME^{N}, DATE_DIMENSION^{D}, NUMBER^{U}, OTHER^{O}, CONTEXT^{C}, NOT_CHECKABLE^{NC} and NON_ENGLISH^{NE}. In this work, we excluded NOT_CHECKABLE^{NC} and introduced a new error category, ADDITION^{A}, 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

[3] https://github.com/google-research-datasets/ToTTo

Category	T5-base (T5-b)		T5-large (T5-l)	
	Count	%	Count	%
No Error	358	47	450	60
Omissions	272	36	218	29
Errors	124	17	86	11
Total Count	754		754	

Table 3: Pilot Study analysis: T5-b and T5-l Models in ToTTo Politics Domain. 'Errors' category is the main focus of our work; 'omissions' are excluded.

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

[4]We downloaded weights from 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.

(a) Input Table with Original Cells highlighted in Yellow

Page Title: 2014 United States Senate elections
Section Title: Elections leading to the next Congress

State	Candidates
Alaska	Dan Sullivan (Republican) 48.0% Mark Begich (Democratic) 45.8% Mark Fish (Libertarian) 3.7% Ted Gianoutsos (Independent) 2.0%
Virginia	Mark Warner (Democratic) 49.1% Ed Gillespie (Republican) 48.3% Robert Sarvis (Libertarian) 2.4%

(b) Input Table with Corrected Cells highlighted in Yellow

State	Candidates	Party	% votes
Alaska	Dan Sullivan	Republican	48.0%
Alaska	Mark Begich	Democratic	45.8%
Virginia	Mark Warner	Democratic	49.1%
Virginia	Ed Gillespie	Republican	48.3%

(c) Outputs Before and After Input Data Correction

T5-base before fix: Alaska: Dan Sullivan (R) defeated Mark Begich (D) 45.8% to^{W} **3.7%**^{U} and Ted Gianoutsos (Independent) 2.0%, and Virginia: Mark Warner won 49.1% to^{W} **2.4%**^{U}.

T5-base after fix: Alaska was won by Dan Sullivan (R) with 48.0% of the vote, and Virginia was won by Mark Warner (D) with 49.1% of the vote.

LLAMA 2-7B before fix: 2014 United States Senate elections were held on November 4^{A}, 2014. **The Republican Party gained control of the Senate for the first time since 2006. The Democrats lost their majority in the Senate and Republicans won a net gain of eight seats to take control of the chamber with a 54-46 majority. The Republicans also took control of the House of Representatives, giving them full control of Congress for the first time since 2010.**^{A}

LLAMA 2-7B after fix: 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%.

Table 4: Fixing Multiple leaders' votes that are not atomic, as detailed in Sec. 5.2.1. 'Before fix' shows the errors produced by models with Table 4a as input and 'After fix' shows both models did not produce any factual error with Table 4b as input. The error tokens in outputs are annotated with respective error code in superscript (W for WORD, U for NUMBER and A for ADDITION errors).

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 Edwin S. Johnson, a Democrat candidate**^{C}.' 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**^{C} error, specifically *when the models made unsupported assumptions*

(a) Input Table with Original Cells highlighted in Yellow

Page Title: Minnesota Amendment 2

Section Title: Results

Choice	Votes	% Votes
Referendum failed No	1,588,752	53.84
Yes	1,362,030	46.16

(b) Input Table with Corrected Cells highlighted in Yellow

Choice	Votes	% Votes
Referendum failed No	1,588,752	53.84
Yes	1,362,030	46.16

(c) Outputs Before and After Input Data Correction

T5-base before fix The Minnesota Amendment 2 was approved[W] by 53.84 percent of voters, with a turnout of[C] 46.16 percent.

T5-base after fix: Minnesota Amendment 2 was defeated by 53.84 percent of voters, while 46.16 percent voted in favor of the measure.

LLAMA 2-7B before fix: 53.84% of Minnesota voters voted for[W] Amendment 2.

LLAMA 2-7B after fix: 53.84% of the voters voted no on Amendment 2, while 46.16% voted yes.%.

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[C] 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[W]. After we included the 'Total time in office' cell and customized the prompt for LLAMA 2, both LLAMA 2 models corrected the WORD[W] 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 WORD^{W} and CONTEXT^{C} errors. NUMBER^{U} 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.

Category	T5-base (T5-b)		T5-large (T5-l)	
	Before	After	Before	After
WORD	62	20	31	12
NAME	13	3	12	2
DATE_DIMENSION	7	1	6	1
NUMBER	12	1	5	0
OTHER	5	2	2	0
CONTEXT	13	3	16	4
ADDITION	2	1	2	1
NON-ENGLISH	20	20	21	21
Total errors	134	**51**	95	**41**
Error reduction	**62%**		**57%**	

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.

- WORD^{W} 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

Category	T5-base (T5-b) baseline		T5-large (T5-l) baseline		LLAMA 2-7B (L-7B)		LLAMA 2-13B (L-13B)	
	Before	After	Before	After	Before	After	Before	After
WORD	27	18	21	12	23	15	22	5
NAME	7	3	6	2	2	1	3	0
DATE_DIM	2	1	1	1	2	0	1	0
CONTEXT	8	3	7	2	6	2	4	0
NUMBER	4	1	1	0	7	1	6	1
OTHER	0	0	0	0	5	0	1	0
ADDITION	0	0	0	0	5	5	7	5
Total errors	48	**26**	36	**17**	50	**24**	45	**11**
Error reduction(%)	**46%**		**53%**		**52%**		**76%**	

Table 7: Individual error count for the 40 challenging samples selected from the previous 210 in Table 6. It shows the comparison of the errors annotated for original data versus corrected data in T5 and LLAMA 2 models.

Category	T5-base (T5-b) baseline		T5-large (T5-l) baseline		LLAMA 2-7B (L-7B)		LLAMA 2-13B (L-13B)	
	Before	After	Before	After	Before	After	Before	After
No error (Higher is better)	0	12	3	16	2	17	5	23
Omissions	0	6	3	8	2	4	4	8

Table 8: Comparison of 'No error' and 'Omissions' unique count for the same 40 samples. i. Increase in 'No error' count indicates error-free outputs after addressing input problems. ii. Some outputs stopped making factual errors after correction but instead omitted part of the input content, resulting in a higher omission count post-fix.

achieved reductions of 33%, 42%, and 34%, respectively.

- Input correction led to a reduction in NAME^{N} and CONTEXT^{C} errors across all models.
- The original input data exhibited more NUMBER^{U} errors in LLAMA 2 models compared to T5, which were significantly reduced after correction. Both LLAMA 2 models completely resolved DATE_DIMENSION^{D} and OTHER^{O} errors.
- ADDITION^{A} 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

ERROR	ALL AGREE	WORD	NAME	DATE_DIM	NUMBER	OTHER	CONTEXT	ADDITION	NO ERROR	TOTAL
WORD	17	0	0	0	2	0	9	3	0	31
NAME	3	0	0	0	0	0	2	0	0	5
DATE_DIM	1	0	0	0	0	0	2	0	0	3
NUMBER	2	2	0	0	0	0	0	0	0	4
OTHER	2	0	0	0	0	0	0	0	1	3
CONTEXT	7	9	2	2	0	0	0	6	0	26
ADDITION	12	1	0	0	0	0	5	0	0	18
NO ERROR	15	0	0	0	0	1	0	0	0	16
Fleiss' kappa overall agreement for three annotators, $k = (pa - pe)/(1 - pe)$ = 0.622										

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.

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-l 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[5]. 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 shows all annotators agreed on the same error type and other columns show how often each annotator selected a different error type. Although the correct token was usually chosen, disagreements primarily occurred in choosing CONTEXT, ADDITION and WORD error types, as shown in red in Table 9. This might be due to the potential similarities in the definitions of CONTEXT errors and ADDITION errors (see App. B.3). While WORD error is comparatively straightforward, one annotator chose CONTEXT errors instead of WORD errors for a few outputs. Cases where an annotator did not mark any errors were in the minority. The inter-annotation agreement on error category classification for 60 outputs, as shown by a Fleiss' kappa of 0.622, indicates substantial agreement between the three annotators.

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.

[5]We release our error annotations from our human evaluation on 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

Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. 2023. Transformers for tabular data representation: A survey of models and applications. *Transactions of the Association for Computational Linguistics*, 11:227–249.

Shuaichen Chang and Eric Fosler-Lussier. 2023. How to prompt llms for text-to-sql: A study in zero-shot, single-domain, and cross-domain settings.

Miao Chen, Xinjiang Lu, Tong Xu, Yanyan Li, Zhou Jingbo, Dejing Dou, and Hui Xiong. 2022. Towards table-to-text generation with pretrained language model: A table structure understanding and text deliberating approach. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8199–8210, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020a. Logical natural language generation from open-domain tables. *CoRR*, abs/2004.10404.

Zhiyu Chen, Wenhu Chen, Hanwen Zha, Xiyou Zhou, Yunkai Zhang, Sairam Sundaresan, and William Yang Wang. 2020b. Logic2Text: High-fidelity natural language generation from logical forms. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2096–2111, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895, Florence, Italy. Association for Computational Linguistics.

Ondřej Dušek, David M. Howcroft, and Verena Rieser. 2019. Semantic noise matters for neural natural language generation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 421–426, Tokyo, Japan. Association for Computational Linguistics.

Dimitra Gkatzia, Oliver Lemon, and Verena Rieser. 2017. Data-to-text generation improves decision-making under uncertainty. *IEEE Computational Intelligence Magazine*, 12(3):10–17.

Javier González Corbelle, Alberto Bugarín-Diz, Jose Alonso-Moral, and Juan Taboada. 2022. Dealing with hallucination and omission in neural natural language generation: A use case on meteorology. In *Proceedings of the 15th International Conference on Natural Language Generation*, pages 121–130, Waterville, Maine, USA and virtual meeting. Association for Computational Linguistics.

Hanxu Hu, Yunqing Liu, Zhongyi Yu, and Laura Perez-Beltrachini. 2023. Improving user controlled table-to-text generation robustness. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2317–2324, Dubrovnik, Croatia. Association for Computational Linguistics.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102, Dublin, Ireland. Association for Computational Linguistics.

Zdeněk Kasner and Ondřej Dušek. 2024. Beyond reference-based metrics: Analyzing behaviors of open llms on data-to-text generation. *arXiv preprint arXiv:2401.10186*.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Steffen C. Pauws, Albert Gatt, Emiel J. Krahmer, and Ehud Reiter. 2018. Making effective use of healthcare data using data-to-text technology. In *Data Science for Healthcare*.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6908–6915.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Vikas Raunak, Arul Menezes, and Marcin Junczys-Dowmunt. 2021. The curious case of hallucinations in neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1172–1183, Online. Association for Computational Linguistics.

Clément Rebuffel, Marco Roberti, Laure Soulier, Geoffrey Scoutheeten, Rossella Cancelliere, and Patrick Gallinari. 2021. Controlling hallucinations at word level in data-to-text generation. *Data Mining and Knowledge Discovery*, 36:318 – 354.

Clément Rebuffel, Laure Soulier, Geoffrey Scoutheeten, and Patrick Gallinari. 2019. A hierarchical model for data-to-text generation. *Advances in Information Retrieval*, 12035:65 – 80.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Sonish Sivarajkumar, Mark Kelley, Alyssa Samolyk-Mazzanti, Shyam Visweswaran, and Yanshan Wang. 2023. An empirical evaluation of prompting strategies for large language models in zero-shot clinical natural language processing. *arXiv preprint arXiv:2309.08008*.

Somayajulu Sripada, Ehud Reiter, Jim Hunter, and Jin Yu. 2002. Sumtime-meteo: Parallel corpus of naturally occurring forecast texts and weather data. *Computing Science Department, University of Aberdeen, Aberdeen, Scotland, Tech. Rep. AUCS/TR0201*.

Yixuan Su, David Vandyke, Sihui Wang, Yimai Fang, and Nigel Collier. 2021. Plan-then-generate: Controlled data-to-text generation via planning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 895–909, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

neural NLG models. In *Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 456–470, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Craig Thomson and Ehud Reiter. 2020. A gold standard methodology for evaluating accuracy in data-to-text systems. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 158–168, Dublin, Ireland. Association for Computational Linguistics.

Craig Thomson, Ehud Reiter, and Somayajulu Sripada. 2020. SportSett:basketball - a robust and maintainable data-set for natural language generation. In *Proceedings of the Workshop on Intelligent Information Processing and Natural Language Generation*, pages 32–40, Santiago de Compostela, Spain. Association for Computational Lingustics.

Craig Thomson, Ehud Reiter, and Barkavi Sundararajan. 2023. Evaluating factual accuracy in complex data-to-text. *Computer Speech & Language*, 80:101482.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Upadhyay and Stewart Massie. 2022. Content type profiling of data-to-text generation datasets. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5770–5782, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Emiel van Miltenburg, Miruna Clinciu, Ondřej Dušek, Dimitra Gkatzia, Stephanie Inglis, Leo Leppänen, Saad Mahamood, Emma Manning, Stephanie Schoch, Craig Thomson, and Luou Wen. 2021. Underreporting of errors in NLG output, and what to do about it. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 140–153, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3544–3552, Online. Association for Computational Linguistics.

Fei Wang, Zhewei Xu, Pedro Szekely, and Muhao Chen. 2022. Robust (controlled) table-to-text generation with structure-aware equivariance learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5037–5048, Seattle, United States. Association for Computational Linguistics.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Fengbin Zhu, Moxin Li, Junbin Xiao, Fuli Feng, Chao Wang, and Tat Seng Chua. 2023. Soargraph: Numerical reasoning over financial table-text data via semantic-oriented hierarchical graphs. In *Companion Proceedings of the ACM Web Conference 2023*, WWW '23 Companion, page 1236–1244, New York, NY, USA. Association for Computing Machinery.

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.

Models	Batch size	Beam Size	Training steps
T5-base	2	10	180,000
T5-large	4	5	9,000

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^{N}

- 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^{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^{N}**, and **Joe Biden^{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[N] instead of Tuesday is a NAME error but May[D] instead of April is a DATE_DIMENSION error.

NUMBER[U]

- 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%**[U] is a NUMBER error.
- Output: The voter turnout was 8,90%, with 10,052 votes. Remarks: The actual turnout was **81.90%**[U]. 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[D]

- 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**[D]. 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[W]

- 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**[W] Al Ramirez and Tim Kalemkarian. Remarks: Fiorina independently defeated all the leaders, so 'along with' is wrong.
- Example Output text: Ling **won**[W] 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[C]

- 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**[C], Michael D. Brown received 62,415 votes while Philip Pannell received 21,552 votes and **write-in candidates**[C] received 1,363 votes. Annotation remarks:
    * **for mayor of Florence Pendleton**[C] - the name of a person is misrepresented as electorate (jurisdiction) in the output.

* write-in candidates[C] - this implies there was more than one write-in candidate.

ADDITION[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[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[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: GSSSDULSVDHSS[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[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[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, DATASIZEMANAGEABLEFORLONGTABLE and IDENTIFYLEADERNAMEORDER 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 CORRECTNONATOMICCELLS, UPDATEHEADERS, ADDRESSMISSINGVALUES, and REPLACESYMBOLS 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 DATASIZEMANAGEABLEFORLONGTABLE($tabularData$) **then**
    return "Please simplify the tabular data with fewer records", null
  end if
  $leaderName, recordedData \leftarrow$ IDENTIFYLEADERNAMEORDER($tabularData, title$)
  $correctionsMade \leftarrow$ **false**
  $correctionsMade \leftarrow$ CORRECTNONATOMICCELLS($tabularData$) **or** $correctionsMade$
  $correctionsMade \leftarrow$ UPDATEHEADERS($tabularData$) **or** $correctionsMade$
  $correctionsMade \leftarrow$ ADDRESSMISSINGVALUES($tabularData$) **or** $correctionsMade$
  $correctionsMade \leftarrow$ REPLACESYMBOLS($tabularData$) **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 DATASIZEMANAGEABLEFORLONGTABLE($tabularData$)
  $maxRows \leftarrow 20$
  $maxCols \leftarrow 10$
  return (length(*tabularData*) $\leq maxRows$) **and** (length(*tabularData*[0]) $\leq maxCols$)
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.

function IDENTIFYLEADERNAMEORDER(*tabularData*, *title*)
  $leaderNameFromTitle \leftarrow$ ExtractLeaderNameFromTitle(*title*)
  $leaderNamesFromRows \leftarrow [\,]$
  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**
    $recordedData \leftarrow [\,]$ ▷ 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"])
  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
      $recordedData \leftarrow$ "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 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.

function CORRECTNONATOMICCELLS($tabularData$)
  $correctionsMade \leftarrow$ **false**
  for all row **in** $tabularData$ **do**
    for all $cell$ **in** row **do**
      if CellIsNonAtomic($cell$) **then**
        $correctedCell \leftarrow$ CorrectCell($cell$) ▷ Separate multiple values into individual columns
        $tabularData[cell] \leftarrow correctedCell$ ▷ Update the corrected cell value
        $correctionsMade \leftarrow$ **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.

function CORRECTHEADERS($tabularData$)
  $correctionsMade \leftarrow$ **false**
  for all row **in** $tabularData$ **do**
    for all $cell$ **in** row **do**
      if cell.header_value $=$ **true then**
        if HeaderIsIncorrect($cell$) **then**
          $correctedHeader \leftarrow$ UpdateHeader($cell$) ▷ Update column and row headers
          $tabularData[cell] \leftarrow correctedHeader$ ▷ Update the corrected header
          $correctionsMade \leftarrow$ **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($tabularData$)
  $correctionsMade \leftarrow$ **false**
  for all row **in** $tabularData$ **do**
    if RowHasMissingValues(row) **then**
      $correctedRow \leftarrow$ FillCellValues(row) ▷ Pass missing cells in the row
      $tabularData[row] \leftarrow correctedRow$
      $correctionsMade \leftarrow$ **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*)
  $correctionsMade \leftarrow$ **false**
  for all $cell$ **in** *tabularData* **do**
    if containsDomainSpecificSymbols($cell$) **then**
      $correctedValue \leftarrow$ substituteSymbolWithEquivalent($cell$) ▷ Replace symbols with words
      $tabularData[cell] \leftarrow correctedValue$ ▷ Update the corrected cell value
      $correctionsMade \leftarrow$ **true**
    else if containsPartyAbbreviations($cell$) **then**
      $correctedValue \leftarrow$ abbreviatePartyNames($cell$) ▷ Abbreviate party names
      tabularData$[cell] \leftarrow correctedvalue$ ▷ Update the corrected cell value
      $correctionsMade \leftarrow$ **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.

(a) Input Table with Original Cells highlighted in Yellow

Page Title: 2012 Cardiff South and Penarth by-election
Section Title: By-election result

Party	**%**	±
Conservative	19.9	8.4
Liberal Democrat	10.8	-11.5

(b) Input Table with Corrected Cells highlighted in Yellow

Page Title: 2012 Cardiff South and Penarth by-election
Section Title: By-election result

Party	**%**	**± % difference with previous election**
Conservative	19.9	8.4
Liberal Democrat	10.8	-11.5

(c) Outputs Before and After Input Data Correction

T5-large before fix: In the 2012 Cardiff South and Penarth by-election, the Liberal Democrats **won**[W] 11.5% of the vote.

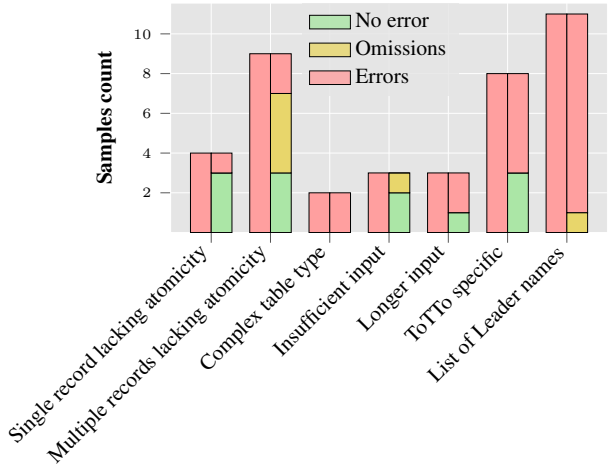
T5-large after fix: In the 2012 Cardiff South and Penarth by-election, the Liberal Democrats **lost by**[W] 11.5%.

LLAMA 2-13B before fix: 2012 Cardiff South and Penarth by-election was **won**[W] by the Liberal Democrats with a **margin**[W] of -11.5%.

LLAMA 2-13B after fix: 2012 Cardiff South and Penarth by-election saw Liberal Democrat candidate lose by 11.5% of the vote compared to the previous election.

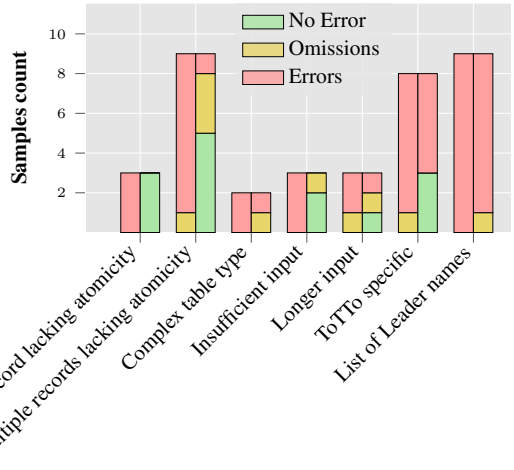
Table 10: ToTTo specific: Swing percentage difference with previous election, as detailed in Sec. 5.2.5. Before fix represents the output generated from input Table 10a and after fix represents the output generated after explicitly changing the header, as shown in input Table 10b.

(a) T5-base: Error Analysis of 40 samples.



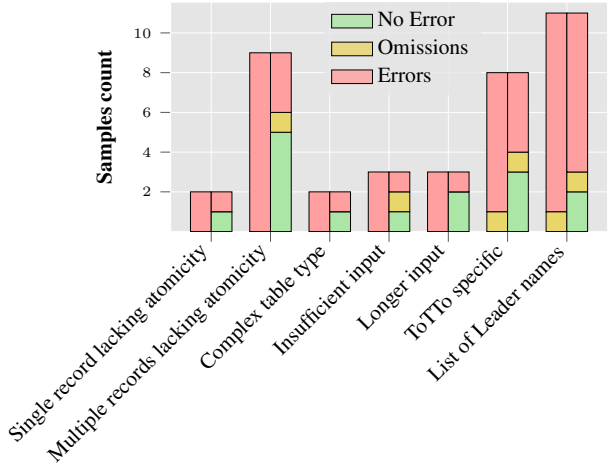
Input Problem Types: before data fix (left bar) versus after data fix (right bar)

(b) T5-large model: Error Analysis of 37 samples.



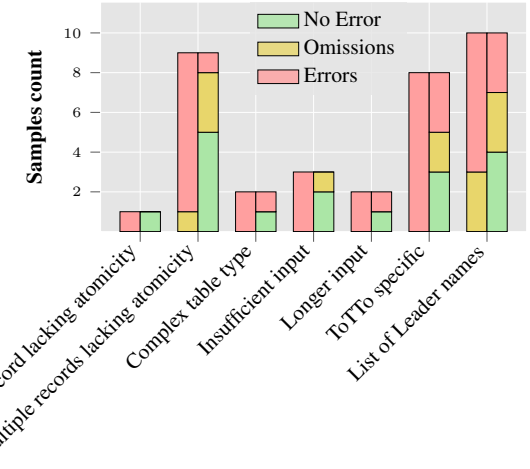
Input Problem Types: before data fix (left bar) versus after data fix (right bar)

(c) LLAMA 2-7B: Error Analysis of 38 samples.



Input Problem Types: before data fix (left bar) versus after data fix (right bar)

(d) LLAMA 2-13B: Error Analysis of 35 samples.



Input Problem Types: before data fix (left bar) versus after data fix (right bar)

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.

(a) Input Table with Original Cells highlighted in Yellow

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

State	Incumbent		Results	Candidates
	Senator	Party		
South Dakota	Coe I.Crawford	Republican	Incumbent lost renomination. New senator elected. Democratic gain.	Edwin S. Johnson (Democratic) 48.32% Charles H. Burke (Republican) 44.47%

(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[C].

T5-large: South Dakota: Republican incumbent Coe I. Crawford lost his bid for the seat to Democratic challenger Edwin S. Johnson[C].

LLAMA 2-7B: 1914 United States Senate elections in South Dakota were held on November 3[A], 1914. Incumbent Republican Senator Coe I. Crawford lost renomination to Democratic candidate Edwin S. Johnson[C]. Johnson defeated Charles H. Dunnigan[N] by a margin of[W] 48.32% to 47.56%[U]. This was a Democratic gain from the previous election.

LLAMA 2-13B: 1914 United States Senate elections were held on November 3[A], 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

State	Incumbent Senator	Incumbent Party	Results	Candidate Senator	Candidate Party	Candidate votes %
South Dakota	Coe I.Crawford	Republican	Incumbent lost renomination. New senator elected. Democratic gain.	Edwin S. Johnson	Democratic	48.32%
South Dakota	Coe I.Crawford	Republican	Incumbent lost renomination. New senator elected. Democratic gain.	Charles H. Burke	Republican	44.47%

(d) Outputs After Input Data Correction

T5-base: In South Dakota, Republican Coe I. Crawford lost renomination to Democrat Edwin S. Johnson[C].

T5-large: South Dakota: Republican incumbent Coe I. Crawford lost his bid for the seat to Democratic challenger Edwin S. Johnson[C].

LLAMA 2-7B: 1914 United States Senate elections were held on November 3,[A] 1914. In South Dakota, Coe I. Crawford, the Republican incumbent Senator lost renomination to Edwin S. Johnson[C], 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[A], 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.

(a) Input Table with Original Cells highlighted in Yellow

Page Title: List of Japanese prime ministers by longevity
Section Title: Longevity of Japanese prime ministers

Prime Minister	Total time in Office	Date of Death	Lifespan
Sanetomi Sanjō	60 days	28 Feb 1891	53 years, 352 days

(b) Input Table with Corrected Cells highlighted in Yellow

Page Title: List of Japanese prime ministers by longevity
Section Title: Longevity of Japanese prime ministers

Prime Minister	Total time in Office	Date of Death	Lifespan
Sanetomi Sanjō	60 days	28 Feb 1891	53 years, 352 days

(c) Outputs Before and After Input Data Correction

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

LLAMA 2-7B before fix: The longest-serving[W] 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[W] 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.*

(a) Input Table with Original Cells highlighted in Yellow

Page Title: Ling Ling Chang
Section Title: 2016 Senate run

California's 29th State Senate district election		
Party	**Candidate**	**% Votes**
Democratic	Josh Newman	50.4
Republican	Ling Ling Chang	49.6

(b) Input Table with Corrected Cells highlighted in Yellow

Page Title: Ling Ling Chang
Section Title: 2016 Senate run

California's 29th State Senate district election		
Party	**Candidate**	**% Votes**
Democratic	Josh Newman	50.4
Republican	Ling Ling Chang	49.6

(c) Outputs Before and After Input Data Correction

T5-base and T5-large before fix: Chang defeated[W] Democrat Josh Newman with 49.6% of the vote.

T5-base and T5-large after fix: Ling Ling Chang was defeated by Josh Newman, a Democrat, with 50.4% of the vote to Chang's 49.6%.

LLAMA 2-13B before fix: Ling Ling Chang won[W] the 2016 Senate run with 49.6% of the vote.

LLAMA 2-13B after fix: Ling Ling Chang, who was running for the Senate seat in California, received 49.6% of votes while Josh Newman, her opponent, received 50.4%.

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.

(a) Input Table with Original Cells highlighted in Yellow

Page Title: Chuck DeVore
Section Title: As U.S. Senate Candidate

Party	Candidate	Votes	%
Republican	Carly Fiorina	1,315,429	56.4
Republican	Tom Campbell	504,289	21.7
Republican	Chuck DeVore	452,577	19.3
Republican	Al Ramirez	42,149	1.8
Republican	Tim Kalemkarian	19,598	0.8

(b) Input Table with Corrected Cells highlighted in Yellow

Page Title: Chuck DeVore
Section Title: As U.S. Senate Candidate

Party	Candidate	Votes	%
Republican	Carly Fiorina	1,315,429	56.4
Republican	Tom Campbell	504,289	21.7
Republican	Chuck DeVore	452,577	19.3
Republican	Al Ramirez	42,149	1.8
Republican	Tim Kalemkarian	19,598	0.8

(c) Outputs Before and After Input Data Correction

T5-base before fix: DeVore **defeated**[W] Republican Tom Campbell with **56.4%**[U] of the vote, **Al Ramirez with 19.3%, and Tim Kalemkarian with 19.3%**[C].
T5-base after fix: DeVore **defeated**[W] Republican nominees Carly Fiorina (56.4%)**, Tom Campbell (21.7%), Al Ramirez (1.8%) and Tim Kalemkarian (0.8%)**[C].

T5-large before fix: DeVore was defeated by Republican Tom Campbell **(56.4%)**[U], **Al Ramirez (19.3%), and Tim Kalemkarian (21.7%)**[C].
T5-large after fix: DeVore was defeated in the U.S. Senate race by Republican nominees Carly Fiorina (56.4%), Tom Campbell (21.7%)**, Al Ramirez (1.8%) and Tim Kalemkarian (0.8%).**[C]

LLAMA 2-7B before fix: Chuck DeVore is a Republican candidate for the U.S. Senate in California. He received 19.3% of the vote in the primary election, which was held on June 8th, 2010. The other candidates were Carly Fiorina (56.4%), Tom Campbell **(24.1%)**[U] and Al Ramirez **(10.2%)**[U]. Tim Kalemkarian **did not run for this office**[C].
LLAMA 2-7B after fix: The number of votes for each leader is as follows: Carly Fiorina - 56.4% Tom Campbell - 21.7% Chuck DeVore - 19.3% Al Ramirez - 1.8% Tim Kalemkarian - 0.8%. Carly Fiorina won the Republican primary for U.S. Senate in California with 56.4 percent of the vote, followed by Tom Campbell at 21.7 percent and Chuck DeVore at 19.3 percent.

LLAMA 2-13B before fix: Carly Fiorina is the leading candidate for U.S. Senate with 56.4% of the vote, followed by Tom Campbell with **20.3%**[U], Chuck DeVore with 19.3%, and Al Ramirez with **4.0%**[U].
LLAMA 2-13B after fix: The Republican Party candidate who won the most votes in the U.S. Senate election was Carly Fiorina with 56.4% of the total votes.

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

Year	Office	Subject	Party	% Votes	Opponent	Party	% Votes
1804	Governor	Joseph Haslet	Republican	48%	Nathaniel Mitchell	Federalist	52%
1807	Governor	Joseph Haslet	Republican	48%	George Truitt	Federalist	52%

(b) Input Table with Corrected Cells highlighted in Yellow

Page Title: Joseph Haslet

Section Title: Almanac

Year	Office	Candidate	Party	% Votes	Opponent	Party	% Votes
1804	Governor	Joseph Haslet	Republican	48%	Nathaniel Mitchell	Federalist	52%
1807	Governor	Joseph Haslet	Republican	48%	George Truitt	Federalist	52%

(c) Outputs Before and After Input Data Correction

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

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

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

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

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

LLAMA 2-13B after fix: 1804: Joseph Haslet (Republican) won[W] 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

Page Title: List of members of the United States House of Representatives in the 67th Congress by seniority
Section Title: List of Representatives by seniority

Rank	Representative	Party	District	Seniority date
Twenty-three non-consecutive terms				
1	Joseph G. Cannon	R	IL-18	March 4, 1915
Fifteen consecutive terms				
2	Frederick H. Gillett	R	MA-2	March 4, 1893
Fourteen non-consecutive terms				
3	Henry A. Cooper	R	WI-1	March 4, 1921
One term				
4	...	...	...	...
5	...	...	...	...
433	...	...	...	...
434	...	...	...	...
435	Adam M. Wyant	PA-22	WI-1	March 4, 1921
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**[C].

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

(c) Input Table with Corrected Cells highlighted in Yellow

Page Title: List of members of the United States House of Representatives in the 67th Congress by seniority
Section Title: List of Representatives by seniority

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

(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.