Differences in Semantic Errors Made by Different Types of Data-to-text Systems

Rudali Huidrom **Anya Belz** **Michela Lorandi**
DCU Natural Language Generation Research Group
ADAPT Research Centre, Dublin City University
Dublin, Ireland
{rudali.huidrom,michela.lorandi,anya.belz}@adaptcentre.ie

Abstract

In this paper, we investigate how different semantic, or content-related, errors made by different types of data-to-text systems differ in terms of number and type. In total, we examine 15 systems: three rule-based and 12 neural systems including two large language models without training or fine-tuning. All systems were tested on the English WebNLG dataset version 3.0. We use a semantic error taxonomy and the brat annotation tool to obtain word-span error annotations on a sample of system outputs. The annotations enable us to establish how many semantic errors different (types of) systems make and what specific types of errors they make, and thus to get an overall understanding of semantic strengths and weaknesses among various types of NLG systems. Among our main findings, we observe that symbolic (rule and template-based) systems make fewer semantic errors overall, non-LLM neural systems have better fluency and data coverage, but make more semantic errors, while LLM-based systems require improvement particularly in addressing superfluous.

1 Introduction

Human evaluation remains the gold standard to determine the quality of texts generated by Natural Language Generation (NLG) systems (van Miltenburg et al., 2023a). One aspect of human evaluation is error analysis, where researchers identify and categorise errors in system outputs. Ideally, it is achieved by manually annotating output text in a multiple-annotators setting (van Miltenburg et al., 2023b). Although labour intensive, error analysis can provide a healthy dose of skepticism and help to ensure systems have the functionality intended (Raji et al., 2022).

Semantic errors, including missing, added or repeated content, are common in current language generation outputs, particularly for neural methods (Kasner and Dušek, 2024). Documenting and analysing these errors in different types of systems helps in understanding specific faults within system output that we can look to address with improved models in a way that per-system scores do not.

In the work reported in this paper, we start by obtaining word-span error annotations of semantic errors in a variety of data-to-text system input/output pairs. We then analyse the annotations to determine how many semantic errors different (types of) systems make, and what specific types of errors they make, and thus to get an overall understanding of semantic strengths and weaknesses among various types of NLG systems. Our specific contributions are as follows:

1. A comprehensive text annotation experiment yielding word span annotations of semantic errors made by a range of different data-to-text systems.
2. The resulting dataset of system outputs with manually annotated semantic errors, providing a basis for valuable insights regarding semantic errors made by different systems.
3. In-depth analysis of the annotated data to identify patterns and correlations between different types of errors.
4. The resulting insights into how NLG system type, input length and new vs. seen inputs relate to specific semantic error types.

The paper is organised as follows. Section 2 presents related work. Section 3 describes the experimental design in detail. Section 4 outlines the overall experiment set-up. Section 5 presents results and analysis. Section 6 offers a discussion. Section 7 concludes with a summary and future directions. The appendices include the participant recruitment email, feedback from pilot participants, annotation steps, and additional results tables and analyses. Data and resources are on GitHub.[1]

[1] RHuidrom96/Differences-in-Semantic-Errors-. . .

Proceedings of the 17th International Natural Language Generation Conference, pages 609–621
September 23–27, 2024.

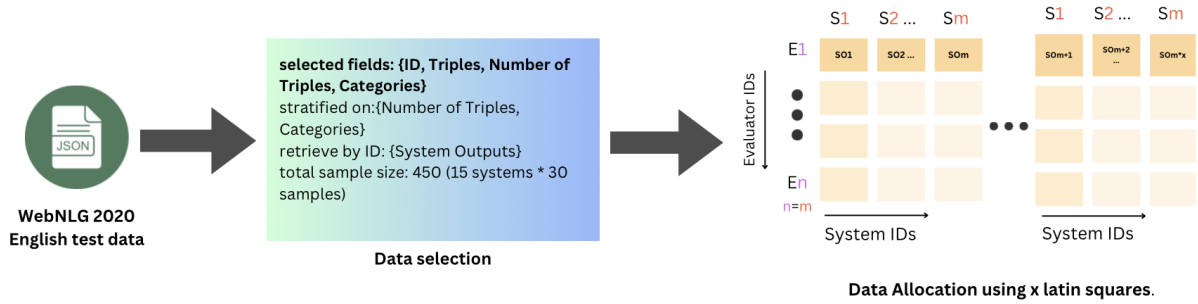

Figure 1: Data Selection and Allocation workflow.

2 Related Work

Many human evaluations of data-to-text systems only score or label outputs at the sentence or paragraph level. If this is all that is known about output quality, finer errors and nuances often go undetected and therefore unaddressed. Reporting word-span level semantic errors found in NLG system outputs is necessary for in-depth error analysis and understanding of the factors contributing to such errors, so that solutions can be tailored to specific error types.

Dušek and Kasner (2020) propose to measure semantic accuracy of data-to-text generation using a neural model pre-trained for natural language inference (NLI). Human annotators used a three-point Likert scale to compare their results to the crowd-sourced human ratings (Shimorina et al., 2018). González-Corbelle et al. (2022) propose an omission and hallucination detector for texts generated by neural data-to-text systems in the meteorology domain, and performed expert analysis with the aim of classifying these errors by severity, taking domain knowledge into account. Li et al. (2023) introduce the Hallucination Evaluation benchmark (HaluEval) to assess hallucination errors in LLMs using human-annotated samples, aiming to improve the models' accuracy in recognising hallucinations. Human annotators used yes/no labels to annotate whether ChatGPT responses contained hallucinated content.

Thomson et al. report different error types in NLG system outputs (Thomson and Reiter, 2020; Thomson et al., 2023). The Shared Task on Evaluating Accuracy (Thomson and Reiter, 2021) focuses on both manual and automatic techniques to evaluate the factual accuracy of texts generated by neural NLG systems. Popovic et al. report error analyses by taking word span into account to evaluate inter-annotator agreement in MT outputs (Popović, 2021; Popović and Belz, 2022). Kasner and Dusek (2024) focus on detecting semantic errors in model outputs by comparing the generated text to the input data. Errors are annotated at word-level, with every word in the output text being considered a potential source of error. This is the most comparable work to ours, although they do not annotate the input since they do not address omission. To the best of our knowledge, none of the other publications report performing word span annotations of semantic errors in input and system output pairs from different data-to-text systems.

3 Experiment Design

3.1 Types of systems

We evaluate a total of 15 data-to-text systems, comprising three rule-based systems and 12 neural systems, of which two are large language models (LLMs) without any training or fine-tuning. 13 systems are from the WebNLG 2020 Shared Task and the other two systems are from Lorandi and Belz (2024). The 13 systems from WebNLG were those that performed best in the shared task based on multiple criteria used in their human evaluation analysis.

Table 1 provides an overview of the 13 systems in terms of their WebNLG categorisation (first column), the name of the participating WebNLG'20 team (where applicable), and the name of the model used by the submitted systems as per the WebNLG'20 system description reports. We moreover colour-code system names by broad system type in orange (rule or template-based), blue (neural non-LLM) and pink (LLM), using inclusive color palettes.[2] This colour scheme will be fol-

[2] https://www.nceas.ucsb.edu/sites/default/files/2022-06/ColorblindSafeColorSchemes.pdf

Categorisation	Participating Team (model type)
Monolingual, mono-task, template-based	[1]RALI (Template-based), [2]DANGNT-SGU (Template-based)
Baseline	[3]Baseline-Forge2020 (Rule-based)
Monolingual, mono-task, neural	[4]TGen (T5), [5]NILC (BART), [6]NUIG-DSI (T5)
Mono-task, bilingual approaches	[7]cuni-ufal (mBART), [8]Huawei Noah's Ark Lab (multilingual transformer-based seq2seq model), [9]OSU Neural NLG (T5), [10]FBConvAI (BART)
Bidirectional, monolingual approaches	[11]Amazon AI (T5), [12]CycleGT (T5)
Bidirectional, bilingual approaches	[13]bt5 (T5)
Large language models, no training or fine-tuning	[14]GPT 3.5, [15]Llama-chat-270B

Table 1: Color-coded (rule-based, non-LLM neural and large language models (LLM)) summary of the participating teams' systems categorisation, taken verbatim from WebNLG 2020 results report.

lowed throughout the paper.

3.2 Data selection and allocation

We randomly selected 450 input-output pairs by stratified sampling based on the number of triples in the input and WebNLG category.[3] We allocate these samples to each evaluator using repeated Latin squares which has the effect that each evaluator annotated a different set of 30 input and system output pairs, and [4] each evaluator assessed two system outputs from each system, given that we used two Latin squares where the size of each Latin square is the number of evaluators by the number of systems (15 x 15). The data selection and allocation process is illustrated in Figure 1.

3.3 Participant selection

We invited researchers at the ADAPT Research Centre (Ireland) to participate in our study via an email (see email template in Appendix A) to the centre-wide mailing list, linking to a sign-up form that asked for English language proficiency (Proficient User – C1, C2, Independent User – B1, B2, Basic User – A1, A2), prior experience with error annotation (yes/no), and an example annotation. Participants were excluded if they had no prior experience with error annotation or if the example was incorrectly annotated. The Google Form used for this purpose is in the supplementary materials on our GitHub.

We received a total of 11 sign-ups. Out of these, 10 marked their English language proficiency as Proficient User (C1, C2), and one marked it as Independent User (B1, B2). Six participants had prior experience with error annotation, while five did not. We selected six participants from the sign-ups based on their prior experience and the correctness of the example annotation. An additional nine participants were selected from a previously conducted pilot experiment (see below); these are proficient users of English and NLP researchers.

3.4 Error categories

We use three error types and definitions for annotation, following Huidrom and Belz (2023). We refrain from using the term 'hallucination' due to its meaning in the field of psychology. For instance, (Blom, 2010) defines hallucination as "a percept, experienced by waking individual, in the absence of an appropriate stimulus from extracorporeal world." Instead, we use the term "addition" as defined below. In the following definitions, 'input' is the set of triples, and 'output' is the verbalisation (text).

- **Omission:** Some content that is present in the input and should be rendered in the output is not present in the output. Moreover, there are no word span(s) in the output that are intended to render it, but do so wrongly. i.e. this type of error can be fixed by adding something to the output.
- **Addition:** The output text contains word span(s) for which there is no corresponding part of the input that they render. In other words, some content that is not present in the input and should not be rendered in the output is nevertheless rendered by some word span(s) in the output. Moreover, there is no content in the input that the word span(s) are intended to render, but render wrongly. i.e. this type of error can be fixed by removing something from the output.
- **Repetition:** Some content is repeated verbatim in the output, but there is no corresponding repetition in the input.

[3]We had intended to also stratify in terms of seen vs. unseen properties, but used the WebNLG'17 list of unseen properties erroneously, so counts aren't in quite the same proportions as the whole dataset.

[4]We chose Latin-square design to optimise cost and benefit.

3.5 Annotation process

We record the word-span annotations of our set of input and system outputs pairs using via the brat annotation tool[5]. The input here is a set of triples, and the system output is the generated verbalisation. Each triple consists of the elements Subject, Predicate, and Object, and the verbalisation. For example, an input triple could be `Take_It_Off (Subject), producer (Predicate), Wharton_Tier (Object)`, and the corresponding system output (verbalisation) could be *Wharton Tiers produced Take It Off*.

The annotation task is to mark and label omissions in the set of input triples, and additions and repetitions in the verbalisation. There can be any number of semantic errors, including none, in any triple-set/verbalisation pair.

3.6 Summarised annotation steps

The following is the summarised annotation steps. Verbatim annotation instructions can be found in Appendix C.

1. *Omission annotation:* The evaluator should check if each element in the input triples is verbalised. If any element is missing, it should be marked as an omission error. If the entire triple is not verbalised, each element of the triple should be marked as an omission. If all elements are verbalised, it means there are no omission errors.
2. *Addition annotation:* The evaluator should check if all content words and phrases in the verbalisation correspond to elements in the triples. If any content word or phrase does not match an element in the triples, it should be marked as an addition error. If all content phrases correspond correctly, it means there are no addition errors.
3. *Repetition annotation:* The evaluator should check for repeated content in the output, including close paraphrases. If any element in the triples is rendered more than once, it should be marked as a repetition error, unless there is corresponding repetition in the input triple elements. If all content words and phrases in the verbalisation correspond correctly to the triples without repetition, it means there are no repetition errors.

[5] https://brat.nlplab.org

4 Human Evaluation

4.1 Data

We use the system outputs from the WebNLG 2020 (Ferreira et al., 2020) on the English test dataset, which contains 1,779 different input triple sets. There are a total of 19 categories in the WebNLG 2020 dataset, of which 16 are present in the training set, and three are unseen in the training set (Film, MusicWork, Scientist). We selected 450 input triple sets with stratification for our experiment. Table 2 shows the overall counts of the number of triples and categories in the WebNLG 2020 English test dataset along with the counts in the stratified samples.

Number of Triples							Categories	
1	**2**	**3**	**4**	**5**	**6**	**7**	**Seen**	**Unseen**
369	349	350	305	213	114	79	966	813
90	90	90	75	60	30	15	285	165

Table 2: Triple size and category counts for the overall dataset (third row) and the stratified sample (fourth row).

4.2 Brat annotation tool setup

We use the brat annotation tool (Stenetorp et al., 2012), a web-based tool for text annotation, to record word-span annotations of semantic errors (omission, addition, and repetition) in input triple sets and system output pairs. We use ngrok[6] to host brat for our experiment. The annotators were provided with the link to the brat annotation tool via email along with login credentials (username and password).

To annotate the errors, the evaluators have to (i) log in to the brat annotation tool using the provided credentials, (ii) select the word span to be marked as an error, which gives a pop-up window containing the list of semantic error types under the 'entity type' label in the interface, (iii) select the correct 'entity type' label for the selected word span, and (iv) log out of the brat annotation tool.

4.3 Pilot experiment and feedback

We conducted a pilot experiment on a set of 10 triples/verbalisation pairs with 10 researchers from ADAPT Research Centre, Ireland. We collected feedback via Google Form to identify questions or issues encountered during the annotation process, and to collect suggestions regarding ways to improve the evaluation design, etc. We paid each evaluator 15 Euros per hour for the pilot.

[6] https://ngrok.com

	System	#Omissions	#Additions	#Repetitions	#Total errors	WebNLG 2020 (Avg. Raw)	
						Fluency	Data Coverage
Rule-based	Baseline-FORGE2020	12	13	2	27	82.430	92.892
	DANGNT-SGU	14	18	1	33	78.594	95.315
	RALI	13	21	2	36	77.759	95.204
Non-LLM neural	Amazon-AI-Shanghai	15	19	0	34	90.286	94.393
	NUIG-DSI	20	14	0	34	88.898	92.063
	NILC	47	36	3	86	74.851	81.605
	TGEN	18	18	0	36	86.163	88.176
	CycleGT	19	14	1	34	84.820	91.231
	FBConvAI	16	23	3	42	90.837	93.169
	OSU-Neural-NLG	11	8	6	25	90.066	95.123
	cuni-ufal	21	15	4	40	87.642	93.291
	bt5	16	19	0	35	88.688	93.836
	Huawei-Noah's-Ark-Lab	30	30	4	64	75.205	84.743
LLM	GPT-3.5	13	26	0	39	-	-
	LLAMA-2 70bchat	24	32	1	57	-	-
	Total error counts	*289*	*306*	*27*	*622*		
	Mean	*19.267*	*20.4*	*1.8*	*41.467*		
	Standard Deviation	*9.177*	*7.763*	*1.859*	*15.95*		

Table 3: Counts of each error type for each system. The last two columns present the average fluency and data coverage scores from the WebNLG'20 human evaluation analysis.

	Error Rate						
Error Type	**1 triple** (n=90)	**2 triples** (n=90)	**3 triples** (n=90)	**4 triples** (n=75)	**5 triples** (n=60)	**6 triples** (n=30)	**7 triples** (n=15)
Omissions	0.167	0.183	0.152	0.23	0.187	0.3	0.2
Additions	0.278	0.139	0.207	0.23	0.217	0.256	0.191
Repetitions	0	0.011	0.015	0.013	0.013	0.055	0.029

Table 4: Rates of omission, addition and repetition errors relative to input size.

One common suggestion was to add more examples to the annotation guidelines, including special cases that annotators should look out for. Other feedback related to how to present the layout of triples/verbalisation pairs on brat, providing step-by-step instructions on using brat, and giving background information on what a triple and verbalisation are. More details can be found in Appendix B.

After improving the evaluation design based on the feedback from our pilot experiment, we conducted our main evaluation study with 15 evaluators on 30 triples/verbalisation pairs for each evaluator. We paid 25 Euros for our main study, estimating that it took about an hour to do. We raised the payment relative to the pilot experiment due to the task's increased complexity in the number of triples/verbalisation pairs to be evaluated. All communication for both the pilot and main experiments took place via email exchanges.

5 Results and Analysis

In this Section, we present our results and analysis. We report the raw error counts (Table 3), and error rates for different input properties (Tables 4, 5, and 6). Lastly, we present further analysis on the correlation between error types and system type.

5.1 Raw error counts

Table 3 provides counts of each error type for each system, including the number of omissions, additions, repetitions, and total errors. Additionally, it includes the average fluency and data coverage scores from the WebNLG'20 human evaluation.

We can see that omission and addition errors are more prevalent and consistent across systems, as indicated by their higher mean values and moderate standard deviations. These errors occur relatively frequently, with less variation between systems, suggesting that their occurrence is more predictable. In contrast, repetition errors occur less frequently but have pronounced relative variability, as evidenced by a standard deviation that exceeds their mean. However, it has to be noted that due to their sparsity, repetition error counts and rates provide a less reliable picture than the other two error types investigated here. Omission and addition errors constitute 46.47% and 49.19% of all errors, respectively, while repetition errors just 4.34%.

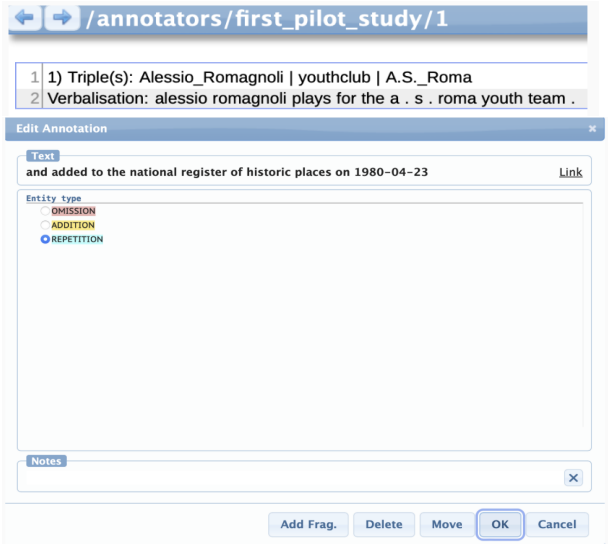

Figure 2: Brat annotation tool interface with example input-output pairs.

Highlighting some system-specific observations, we can observe that (i) NUIG-DSI, a non-LLM neural system, has a higher proportion of omission errors compared to the other two error types (58.82%); (ii) GPT-3.5 (LLM) shows a higher proportion of addition errors (66.67%) and has no repetition errors; (iii) OSU-Neural-NLG, a non-LLM neural system has a relatively high proportion of repetition errors (24%); and (iv) cuni-ufal, another non-LLM neural system, also has a high proportion of repetition errors (10%).

Rule-based systems have fewer total errors on average than neural systems. However, rule-based systems have a higher tendency towards addition errors, suggesting they struggle with filtering out unnecessary items. Non-LLM neural systems, show a balanced distribution between omission and addition errors. Repetition errors are relatively low across all non-LLM neural systems, except for OSU-Neural-NLG, which has higher repetition rates (24%). LLM-based systems are observed to have a strong tendency to add extra content but manage to avoid repetitions effectively.

The last two columns in Table 3 present fluency and data coverage scores copied verbatim from the WebNLG 2020 Shared Task human evaluation, derived from the WebNLG 2020 Human Evaluation test set. Systems with higher fluency scores tend to have fewer total errors, especially omission and repetition errors. For example, Amazon-AI-Shanghai, FBConvAI and OSU-Neural-NLG have fluency scores above 90 and these systems show similarly high levels of addition and omission errors, except for FBConvAI which has relatively higher rate of addition errors.

Systems with high data coverage tend to have higher addition errors. For example, DANGNT-SGU, Amazon-AI-Shanghai and OSU-Neural-NLG have data coverage score above 94 and these systems exhibit low omission errors but sometimes have more additions as in DANGNT-SGU. Meanwhile, low fluency and low data coverage systems have higher errors across all types, in general. For example, NILC have the lowest fluency (74.851) and data coverage (81.605) score and highest total errors (86), suggests that low fluency and low data coverage correlates with higher errors, especially omission and addition errors. While specific fluency and data coverage scores are not available for LLM-based systems, the error patterns suggest a tendency for over-generation (addition errors).

Overall, the rule-based systems are more consistent and generally reliable with balanced errors, meaning that the rule-based systems tend to have a more uniform error distribution, with less variation in the number of omission, addition, and repetition errors between the different rule-based systems. Non-LLM neural systems can achieve higher fluency and data coverage but need careful management of errors, meaning that high fluency and high data coverage correlates with lower errors. LLM-based systems show potential but require improvement in addressing over-generation (additions) and missing content (omissions) issues effectively.

5.2 Error rates relative to different factors

In this section, we calculate error rates relative to (i) **input size** (number of triples); (ii) **system type** (rule/template-based, non-LLM neural, LLM-based); and (iii) **seen vs. unseen properties**, in order to gain a better understanding of how these factors relate to errors.

Rates of omission, addition and repetition errors relative to input size. Table 4 shows occurrence rates for omission, addition and repetition errors relative to different numbers of input triples (1–7). We define these error rates as:

$$\text{Error Rate}_{input\ size} = \frac{E_{i,e}}{i \times T_i} \quad (1)$$

where e denotes the error type (one of omission, addition, and repetition), and i denotes input triple size (one of 1–7). $E_{i,e}$ is the number of errors found for the given error type e and input size i, while T_i is the total number of data items of length

i. Multiplying T_i by i gives us the total number of triples in data items of input size i. Intuitively, these error rates thus capture how many e.g. omission errors there are per triple for a given input size. Note that we need to look at per-triple rates here to be able to compare error rates across input sizes. For consistency, we also report the other two error rates below per triple.

None of the error types follow a uniformly increasing or decreasing trend according to Table 4. Omissions and repetitions have a slightly clearer tendency to increase with more triples, indicating greater challenges in handling larger input sizes. Notably, 4-triple inputs show the same error rate (0.23) in both omissions and additions, this being the highest observed error rate in omissions. Repetitions follow a clearer upward trend with increasing numbers of triples, although they remain the least frequent error type.

It is clear from Table 4 that the complexity introduced by higher numbers of triples impacts error rates to some extent, and this is clearer in the case of omissions and repetitions. Additions do not show any clear trend with changing input sizes.

Rates of omission, addition and repetition errors relative to system type. Second, we look at occurrence rates for omission, addition and repetition errors for rule-based, non-LLM neural and LLM system types. We define this error rate as follows:

$$\text{Error Rate}_{system\ type} = \frac{E_{s,e}}{i_s \times T_s} \quad (2)$$

where e denotes the error type (one of omission, addition and repetition), and s denotes system type (one of rule-based, non-LLM neural and LLM). $E_{s,e}$ represents the number of errors found for the given error type e and system type s. T_s is the total number of data items produced by systems of type s, and i_s is the average number of input triples in data items of type s.

Error Type	Error Rate			
	Rule-based	Neural		
		LLM + Non-LLM neural	LLM	Non-LLM neural
Omissions	0.137	0.219	0.195	0.224
Additions	0.182	0.223	0.305	0.206
Repetitions	0.018	0.019	0.005	0.022

Table 5: Rates of omission, addition and repetition errors relative to system type.

Table 5 highlights substantial differences in error rates between the different types of system. Rule-based systems have the lowest omission rate (0.137), with non-LLM neural systems having the highest (0.224), and LLM systems (0.195) falling in between. The indication is that overall, neural architectures are more prone to omission errors than rule-based systems, although LLMs less so than other neural systems.

Overall, addition rates are higher than omission rates, except for non-LLM neural systems. The gap is particularly big for LLM systems which also have the highest overall addition rate (0.305); rule-based systems have the lowest (0.182).

Repetition rates are notably low across all systems. LLM systems have the lowest repetition rate (0.005), suggesting a superior ability to avoid redundancies. Non-LLM neural systems have the highest repetition rates (0.022), followed closely by rule-based systems (0.018).

Rule-based systems generally show lower error rates in both omissions and additions compared to neural systems, suggesting a more controlled and predictable output. Non-LLM neural systems have lower addition error rates (0.206), but these are still higher than those of rule-based systems. LLM models, while showing high error rates in additions, perform well in minimising repetition errors.

Rates of omission, addition and repetition errors relative to seen/unseen category. Finally, we look at occurrence rates for omission, addition and repetition errors relative to seen vs. unseen properties. We define this error rate as:

$$\text{Error Rate}_{seen/unseen} = \frac{E_{c,e}}{i_c \times T_c} \quad (3)$$

where e denotes error type (one of omission, addition and repetition), and c denotes category (one of seen and unseen). $E_{c,e}$ is the number of errors found for the given error type e and category c. T_c is the total number of data items in category c produced by systems, and i_c is the average number of input triples in data items of type c.

Error Type	Error Rate	
	Seen (size 1-6 only)	Unseen
Omissions	0.144	0.301
Additions	0.199	0.246
Repetitions	0.011	0.03

Table 6: Rates of omission, addition and repetition errors relative to seen vs. unseen category.

Table 6 shows the resulting error rates. Note that error rates are computed on the subset of data items

of input lengths 1–6, because that is all we have for the seen category. We observe that the omission rate for data items containing unseen properties (0.301) is more than twice that of data items with only seen properties (0.144). This suggests that when the systems encounter data it has previously been exposed to, it is much better at ensuring that necessary elements are not omitted. For addition rates, the difference between items with seen (0.199) and unseen (0.246) properties is smaller, Repetition errors are the least frequent across both categories, with 0.011 for seen and 0.03 for unseen, but here nearly three times as many mistakes are made for unseen properties.

5.3 Correlation between error types by system type

In Table 7, we report Pearson's correlation coefficients between error types for all system types combined (last row), and separately by system type (rest of table).

	Om vs Add	**Add vs Rep**	**Rep vs Om**
Rule-based	0.619	-0.143	-0.866
LLM	NA	NA	NA
Non-LLM neural	0.847	0.046	0.152
All neural	0.712	-0.076	0.197
Overall correlation	*0.715*	*-0.068*	*0.192*

Table 7: Pearson's correlation coefficients for pairs of error types, separately for the three system types at the top (NA for LLMs where we only have two data points), and for all system types in the last row (*Overall correlation coefficient*). Om=Omission, Add=Addition, Rep=Repetition.

We observe a strongly positive overall correlation between omissions and additions (0.715), i.e. systems that make more omission errors also tend to make more addition errors. In contrast, there is no correlation between additions and repetitions (-0.068), or between repetitions and omissions (0.192), when not differentiating between systems.

However, when looking at correlations for system types separately, rule-based systems show a strong inverse relationship between repetitions and omissions (-0.866). Both non-LLM neural and all neural systems show strong positive correlations between omissions and additions (0.847 for the former, and 0.712 for the latter). Non-LLM neural systems have the highest correlation between omissions and additions.

6 Discussion

Correlation and Dependency Insights. We observe a strong positive correlation between omissions and additions across different types of systems, perhaps indicating a common underlying cause for these errors where they do occur. Notably, neural systems (both LLM and non-LLM) exhibit this trend, perhaps suggesting that when these systems fail to include expected elements, they overcompensate by adding unexpected ones.

Distinct System Type Behaviours. Rule-based systems show a strong negative correlation between repetitions and omissions, and have a higher tendency towards addition errors than the other two, possibly because they struggle with precision in filtering out unnecessary items despite their intended factual accuracy (Gatt and Krahmer, 2018).

On the other hand, the neural systems all have strong positive correlations between omissions and additions. Non-LLM neural systems show the highest such correlation, emphasising the need for robust training and error-mitigation strategies.

Impact of Seen vs. Unseen Data. All error rates are higher in data containing unseen properties than in data containing only seen. We observe a clear trend where systems perform better on familiar (seen) data across all error types. This is consistent with the expectation that models or systems are generally more accurate when dealing with data they have previously encountered. The considerably higher error rates for the unseen category indicate that systems' cannot transfer all learning to unseen data. This is particularly evident in the substantial increase in omission and repetition errors, suggesting that the underlying model may require further training or fine-tuning to improve its generalisation capabilities. In contrast, addition errors show a smaller increase.

Errors by Input Complexity (Number of Triples) We observe that omissions and repetition rates have an overall tendency to increase with more triples, indicating that handling larger input sizes presents greater challenges. The fluctuation in addition rates without a clear trend suggests that this error type might be affected by specific characteristics of the input data rather than its size alone.

7 Conclusion and Future Work

In the work presented in this paper, we conducted a manual word-span annotation experiment with the aim of investigating the different types and numbers of semantic errors observed in the texts generated by 15 table-to-text generation systems, namely 13 WebNLG 2020 systems and two more recent LLM-based systems. We have described the error types, instructions for the evaluation and set up of experiments we used for this purpose. We have presented an analysis of the absolute numbers of errors made by different systems, and the error rates observed relative to input size, system type and unseen vs. seen properties.

Among our findings, we observed high correlation between omission and addition errors, higher correlations between omission and addition errors in neural systems, and higher error rates in the unseen category compared to seen for for all error types. Overall, we found that the symbolic (rule and template-based) systems are more semantically consistent with the input. Non-LLM neural systems achieve higher fluency and data coverage but need careful management of semantic errors, while LLM-based systems require improvement particularly in addressing over-generation (additions) and missing content (omissions). Among these results the particularly high addition error rate of LLM systems (0.305) stands out. These observations pinpoint future directions for what to focus on in improving output quality in different types of systems.

Ethics Statement

Our experiments were conducted with the approval of the university's Research Ethics Committee Board. We adhered to the structure of the ARR responsible research checklist and ensured the anonymity of all participants. The risk associated with this study was minimal.

Acknowledgements

We extend our sincere gratitude to all the evaluators who took part in this study. Special thanks to Craig Thomson, Simon Mille, Mohammed Sabry and Alessandra Mileo for their insightful feedback and comments during the study. We are also grateful for the helpful feedback and advice from the anonymous reviewers.

Huidrom's work is supported by the Faculty of Engineering and Computing, DCU, via a PhD studentship. Lorandi's work was conducted with the financial support of the Science Foundation Ireland Centre for Research Training in Digitally-Enhanced Reality (d-real) under Grant No. 18/CRT/6224. All authors benefit from being members of the SFI Ireland funded ADAPT Research Centre. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

Jan Dirk Blom. 2010. *A dictionary of hallucinations*. Springer.

Ondřej Dušek and Zdeněk Kasner. 2020. Evaluating semantic accuracy of data-to-text generation with natural language inference. *arXiv preprint arXiv:2011.10819*.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris Van Der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 bilingual, bi-directional webnlg+ shared task overview and evaluation results (webnlg+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

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.

Rudali Huidrom and Anya Belz. 2023. Towards a consensus taxonomy for annotating errors in automatically generated text. In *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, pages 527–540, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

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*.

Zdeněk Kasner and Ondrej Dusek. 2024. Beyond traditional benchmarks: Analyzing behaviors of open LLMs on data-to-text generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12045–12072, Bangkok, Thailand. Association for Computational Linguistics.

Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. Halueval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6449–6464.

Michela Lorandi and Anya Belz. 2024. High-quality data-to-text generation for severely under-resourced languages with out-of-the-box large language models. *arXiv e-prints*, pages arXiv–2402.

Maja Popović. 2021. Agree to disagree: Analysis of inter-annotator disagreements in human evaluation of machine translation output. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 234–243.

Maja Popović and Anja Belz. 2022. On reporting scores and agreement for error annotation tasks. In *Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 306–315.

Inioluwa Deborah Raji, I Elizabeth Kumar, Aaron Horowitz, and Andrew Selbst. 2022. The fallacy of ai functionality. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 959–972.

Anastasia Shimorina, Claire Gardent, Shashi Narayan, and Laura Perez-Beltrachini. 2018. *WebNLG challenge: Human evaluation results*. Ph.D. thesis, Loria & Inria Grand Est.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107.

Craig Thomson and Ehud Reiter. 2020. A gold standard methodology for evaluating accuracy in data-to-text systems. *arXiv preprint arXiv:2011.03992*.

Craig Thomson and Ehud Reiter. 2021. Generation challenges: Results of the accuracy evaluation shared task. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 240–248, Aberdeen, Scotland, UK. Association for Computational Linguistics.

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

Emiel van Miltenburg, Anouck Braggaar, Nadine Braun, Debby Damen, Martijn Goudbeek, Chris van der Lee, Frédéric Tomas, and Emiel Krahmer. 2023a. How reproducible is best-worst scaling for human evaluation? a reproduction of 'data-to-text generation with macro planning'. *Human Evaluation of NLP Systems*, page 75.

Emiel van Miltenburg, Miruna Clinciu, Ondřej Dušek, Dimitra Gkatzia, Stephanie Inglis, Leo Leppänen, Saad Mahamood, Stephanie Schoch, Craig Thomson, and Luou Wen. 2023b. Barriers and enabling factors for error analysis in nlg research. *Northern European Journal of Language Technology*, 9(1).

A Participants Recruitment Email

The following is our email template that we sent to recruit participants for our experiment.

Subject: Participants needed for data-to-text system evaluation (1 Hour, 25 Euros)

Dear all,
I hope this email finds you well. My name is [FirstName LastName], and I am currently working on a project focusing on the human evaluation of data-to-text system outputs as a part of my PhD thesis. I am reaching out to you to invite you to participate in this exciting research opportunity.

The aim of this project is to evaluate semantic errors (addition, omission, substitution, repetition) in the input (RDF triples) and data-to-text system outputs pairs from WebNLG 2020 Shared Task. We would need evaluations to be completed no later than [DD MM YY].

Our pilot experiment showed that the evaluation should take about an hour and we are offering 25 Euro for this task.

Prior to the evaluation process, there will be a training session to familiarise the participants with the annotation tool we will be using and of course, provide clear guidelines on how to evaluate these system outputs. We will ensure that the participants have all the necessary resources and support to carry out the evaluation effectively. To acknowledge the time and effort, we are offering compensation for your participation.

We believe that this research project makes a significant contribution to the scientific work in the field.

If you are interested in being a part of this research project and contributing to the field, please express your interest by filling out this Google Form.

Thank you for considering this opportunity. Your participation is highly valued, and I look forward to the possibility of working together on this important research project.

Best regards,
[Signature]

B Pilot Participants Feedback

Nine out of 10 evaluators filled in the reflection form. The other evaluator gave their feedback via text communication. In this section, we report the feedback as received from the reflection form. We summarise them below:

- Five out of nine evaluators expressed the overall pilot experiment was *neither easy nor difficult*, two of them marked as *easy* and the other two marked it as *difficult*.
- Six out of nine evaluators found the annotation guidelines *easy* to follow, two of them marked it as *neither easy nor difficult* and one of them found to be *difficult*.
- On an average, it took about 20 minutes for the evaluators to understand the annotation guidelines.
- On an average, it took about 25 minutes for the evaluators to complete the annotation task.
- All evaluators confirmed that they read the annotation guidelines before starting the annotations.
- Six out of nine evaluators found the brat annotation *difficult* to use. Meanwhile, three of them found it *easy*.
- Seven out of nine evaluators expressed their need on more training for using the brat annotation tool (apart from Section 3 "Instructions for using the brat annotation tool" in the instructions document) whereas two of them answered a no.
- Seven out of nine evaluators found the error type's definitions and examples easy to follow in the instructions document *easy* to follow. Meanwhile, two of them found it *difficult*.
- Eight out of nine participants expressed their interest in the main study, one of them expressed as a maybe.

C Annotation Steps

We asked annotators to follow the following annotation steps, as part of the annotation guidelines:

1. In the first step, the evaluator should examine whether each element in the input triples is verbalised or not. If an element is not expressed in the verbalisation, mark the element as an omission error type in the triple.

   If the whole triple is not expressed in the verbalisation, mark each element as an omission error type in the triple. For example, if the triple 'ENAIRE | city | Madrid' is not expressed in the verbalisation, then mark 'ENAIRE' as an omission, 'city' as an omission and 'Madrid' as an omission.

   If each element in the input triples is verbalised which means there is no omission error, then proceed to the second step.
2. In the second step, the evaluator should examine whether all the content words and phrases in the verbalisation render a corresponding element(s) in the triples.

   If a content word phrase does not render a corresponding element in the input triples, mark it as an addition error type.

   If all the content phrases in the verbalisation render a corresponding element in the input triples this means there is no addition error, so proceed to the third step.
3. In the third step, the evaluator should check if any part of the output is repeated, including close paraphrases. This is the case e.g. if an element in the triples is rendered more than once. If there is a content phrase that is repeated in this sense, mark it as a repetition error type.

   If all the content phrases in verbalisation include all the elements in the triples without an extra in the verbalisation that has no relation in the input triples, which means there is no repetition error, then proceed to the next pair of triple(s) and verbalisation.

D Additional Notes Given to Evaluators

We provide the following notes below to the evaluators along with the annotation guidelines. More details in Appendix E.

- If there is more than one triple in the input, triples are enclosed within single quotes

(‘ ’) and separated by commas. For example, ‘Joe_Biden | president | United_States’, ‘Joe_Biden | birthPlace | Pennsylvania’.

- The evaluator should be careful while selecting the word span when marking an error. The evaluator should select complete tokens, i.e., words in the text, that are delimited by whitespace.

  For example, the selection for ‘president’ in ‘Joe Biden is the president of the United States.’ is correct, but selecting just ‘pres’ is not correct, as in ‘Joe Biden is the president of the United States.’ Similarly, ‘Joe_Biden | president | United_States’ is correct, but ‘Joe_Biden | president | United_States’ is not.

- The evaluator should consider the inferred verbs and tenses correct in verbalisations as long as they are implied by the information in the input triple(s).

  For example, consider the input triple “Alessio_Romagnoli | youthclub | A.S._Roma” and the corresponding verbalisation “alessio romagnoli plays for the a . s . roma youth team.” Here ‘plays for’ can be inferred from the presence of ‘youthclub’ in the input triple. This is considered valid/correct and should not be marked as an error.

- However, cases such as, ‘youthclub’ being verbalised as ‘youthteam’ (‘youthclub’ is not rendered in the output and ‘youthteam’ is added in the output) or ‘AC_Hotel_Bella_Sky_Copenhagen’ verbalised as ‘hotel bella sky copenhagen’ (‘AC_Hotel_Bella_Sky_Copenhagen’ should be marked as an omission and ‘hotel bella sky copenhagen’ as addition) should be marked as errors.

- The evaluator should take extra care with units, dates and other numerical values and their conversions. For example, if ‘1234 m’ is verbalised as ‘1.234 km’ then it should not be considered an error. If ‘2006-12-31’ is verbalised as ‘31st July 2016’ then it should be marked as an omission (‘2006-12-31’ is not rendered in the output), and addition (‘31st July 2016’ is added in the output). If ‘610.0’ is verbalised as ‘610 metres’ then it should be considered an error where ‘metres’ will be an addition error.

E Other Supplementary Materials

We have also included our participation selection form, participation reflection form and annotation guidelines as a part of the supplementary materials for this paper. We share all data and other resources on our GitHub link here: RHuidrom96/Differences-in-Semantic-Errors-Made-by-Different-Types-of-Data-to-text-Systems.

F Tables

	System	#With Error	#Error Free
Rule-based	Baseline-FORGE2020	15	15
	DANGNT-SGU	12	18
	RALI	13	17
Non-LLM neural	Amazon-AI-Shanghai	13	17
	NUIG-DSI	10	20
	NILC	21	9
	TGEN	14	16
	CycleGT	14	16
	FBConvAI	15	15
	OSU-Neural-NLG	10	20
	cuni-ufal	13	17
	bt5	14	16
	Huawei-Noah’s-Ark-Lab	22	8
LLM	GPT-3.5	16	14
	LLAMA-2 70bchat	18	12

Table 8: Counts of each *with error* and *error free* sample for each system.

Table 8 summarises the performance of various systems in terms of the number of *with error* and *error free* samples. Each system has a total of 30 samples. The distribution of *with error* versus *error free* samples varies across the systems, with no system being completely error-free.

System Type	Average Error Rate per System
Rule-based	0.44
Non-LLM neural	0.49
LLM	0.57

Table 9: Average Error Rates per System Type for samples with errors

Table 9 presents average error rates for samples containing errors across different system types. The formulas for calculating these average error rates per system type are detailed in equations 4 and 5. Rule-based systems exhibit the lowest average error rate of 0.44. In comparison, Non-LLM neural systems have an average error rate of 0.49. LLM systems, on the other hand, demonstrate the highest average error rate of 0.57. This summary highlights how different system types perform in terms of error rates, providing insight into their relative effectiveness.

$$\text{Average Error Rate per System Type} = \frac{\sum(\text{Error Rate per System})}{\text{Number of Systems in the System Type}} \tag{4}$$

$$\text{Error Rate per System} = \frac{\text{Number of Samples with Errors}}{\text{Total Number of Samples}} \tag{5}$$