Are Multi-Agents the new Pipeline Architecture for Data-to-Text Systems?

Chinonso Cynthia Osuji[♡,♣], **Brian Timoney**[♣], **Mark Andrade**[♡],
Thiago Castro Ferreira[◇], **Brian Davis**[♡,♣]
[♡] ADAPT Research Centre, Ireland, [♣] Dublin City University, Ireland
[◇] Fluminense Federal University, Brazil
firstname.lastname@adaptcentre.ie, brian.timoney3@mail.dcu.ie,
thiago.ferreira@gmail.com

Abstract

Large Language Models (LLMs) have achieved remarkable results in natural language generation, yet challenges remain in data-to-text (D2T) tasks, particularly in controlling output, ensuring transparency, and maintaining factual consistency with the input. We introduce the first LLM-based multi-agent framework for D2T generation, coordinating specialized agents to produce high-quality, interpretable outputs. Our system combines the reasoning and acting abilities of ReAct agents, the self-correction of Reflexion agents, and the quality assurance of *Guardrail* agents, all directed by an *Orchestrator* agent that assigns tasks to three specialists—*content ordering*, *text structuring*, and *surface realization*—and iteratively refines outputs based on *Guardrail* feedback. This closed-loop design enables precise control and dynamic optimization, yielding text that is coherent, accurate, and grounded in the input data. On a relatively simple dataset like WebNLG, our framework performs competitively with end-to-end systems, highlighting its promise for more complex D2T scenarios.

1 Introduction

Data-to-text (D2T) generation is the process of converting structured data, like meaning representations or knowledge graphs, into coherent and fluent natural language text (Gatt and Krahmer, 2018; Reiter and Dale, 2000). Traditional D2T systems typically employ rule-based methods, using explicitly defined rules or templates to generate textual outputs (Reiter and Dale, 2000; Mille et al., 2023, 2024a). Although rule-based systems offer transparency and interpretability, they face challenges with respect to scalability and adaptability to new or diverse domains and data types (Heidari et al., 2021). In contrast, end-to-end (E2E) neural architectures achieve impressive results, but limited supervision during generation makes their processes largely opaque (Gatt and Krahmer, 2018; Ferreira et al., 2019).

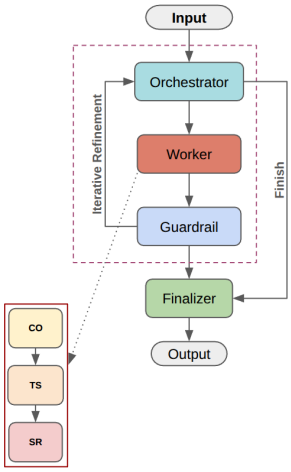

Figure 1: Data-to-text Multi-Agent Framework.

Modern pipeline architectures offer a partial solution to these challenges by decomposing the generation task into distinct modular phases, including content selection (choosing the relevant data), planning (organizing this data logically), and surface realization (constructing the final output text) (Reiter and Dale, 2000). Each phase can independently utilize various methods, such as rule-based, statistical, or neural approaches (Mille et al., 2017; Lapalme, 2024; Ferreira et al., 2019; Cunha et al., 2024; Osuji et al., 2024b,a). Despite providing modularity and transparency, neural pipeline systems still struggle with complex or ambiguous inputs and usually lack integrated mechanisms for error detection and correction, a shortcoming highlighted by concepts such as self-monitoring through feedback loops and revision-based architectures (Pickering and Garrod, 2013; Gatt and Krahmer, 2018).

To overcome these limitations, we introduce a multi-agent framework that segments the D2T generation process into specialized subtasks, each managed by a dedicated large language model (LLM)-powered agent. At the center of this framework is a supervisor (orchestrator) agent, which directs three worker agents tasked with *content ordering*, *text structuring*, and *surface realization*. After each

Proceedings of the 18th International Natural Language Generation Conference, pages 542–553
October 29–November 2, 2025.

stage, the outputs are rigorously evaluated by a set of inspection agents—structured as Guardrails (Rebedea et al., 2023) and designed to assess essential quality aspects, including fluency, semantic adequacy, and ensuring fidelity to the input data.

When errors are detected, guardrails provide targeted feedback that the orchestrator incorporates into subsequent prompts for correction. This dynamic and feedback-driven process draws inspiration from recent innovations such as Reflexion (Shinn et al., 2023), ReAct (Yao et al., 2023), and Self-Refine (Madaan et al., 2023), which have demonstrated the value of combining reasoning, acting, and reflective iteration. As a recent addition to the D2T field, this multi-agent framework opens up new possibilities for improving textual fidelity and control through its iterative, inspectable, and role-oriented architecture. The code and results are publicly available[1].

2 Related Work

Traditional approaches to D2T generation, including early pipeline architectures (Reiter and Dale, 2000), segment the generation process into modular stages. Although interpretable, these systems suffer from error propagation and lack flexibility. End-to-end neural models, particularly those powered by LLMs (Radford et al., 2019; Brown et al., 2020; OpenAI, 2023), offer impressive performance, but their generation processes are typically opaque and challenging to audit when errors occur.

Multi-agent systems have recently emerged as a promising solution to combine modularity with the expressive power of LLMs (Guo et al., 2024; Xi et al., 2025). These systems simulate collaborative problem solving by assigning different roles to agents and enabling structured communication between them. Techniques like ReAct (Yao et al., 2023) combine verbal reasoning and actions, while Reflexion (Shinn et al., 2023) enables agents to iteratively self-improve using natural language feedback. *Guardrails* or inspection agents further enhance quality control by enforcing constraints on output behavior.

The evaluation process in multi-agent systems also benefits from structured feedback. ChatEval (Chan et al., 2024) and other debate-style evaluators (Du et al., 2024) use agent discussion to reach consensus on generation quality, highlighting the importance of diverse roles and feedback loops. Parameter-free optimization methods leverage prompt engineering and agent reflection without fine-tuning, enabling lightweight adaptation (Du et al., 2025).

Our framework synthesizes these developments by integrating role-based agent specialization, feedback-driven orchestration, and robust evaluation guardrails into a single D2T generation pipeline. To the best of our knowledge this is the first attempt to apply a multi-agent system to a D2T task.

3 Methodology

This study uses the English Enriched WebNLG 2020 test set (Castro Ferreira et al., 2020), a widely adopted benchmark for data-to-text generation. This version includes sub-task annotations, which guided both the design of worker roles and the development of guardrail evaluation prompts. The dataset comprises 1,779 sets of RDF triples paired with human-written reference texts, requiring models to verbalize each set as coherent and natural-sounding text. WebNLG covers multiple domains and a variety of linguistic styles, supporting comprehensive evaluation of factual coverage and generation fluency. We make use of zero temperature settings for LLMs used for this study. Full prompt templates and human evaluation guidelines are provided in Appendix 5.

3.1 Multi-Agent System

As shown in Figure 1, our system is organized into **four** core modules: i) an **Orchestrator Agent**, ii) *three* specialized **Worker Agents**—*Content Ordering (CO)*, *Text Structuring (TS)*, and *Surface Realization (SR)*, iii) a set of task-specific **Guardrail Agents**, and a iv) **Finalizer** which produces the final validated text output. All agents in our workflow use the GPT-4.1 (OpenAI, 2025a) language model as the underlying engine, operating with a temperature setting of zero. The workflow consists of:

i. Orchestrator: The Orchestrator Agent supervises the workflow by breaking down the data-to-text (D2T) generation task into smaller subtasks, which are then dynamically assigned to the relevant Worker Agents. The Orchestrator operates with access to the user prompt, its recent interaction histories between workers, and guardrail feedback if any. Drawing inspiration from hierarchical communica-

[1] https://github.com/NonsoCynthia/Data-to-text-Agent

tion in LLM-based multi-agent systems (Guo et al., 2024), the Orchestrator not only delegates tasks but also adapts instructions in response to guardrail evaluations. Through iterative prompt refinement at each stage, it ensures that the 3-stage pipeline advances toward producing coherent, accurate and validated output.

ii. Worker Agents: Because WebNLG triples already encode interpreted facts, our system begins at the document planning stage. Building on prior work (Osuji et al., 2024b,a) that utilized a 3-stage pipeline—and also observed that reducing the number of stages in the pipeline can improve results—we structure our framework around three specialized Worker Agents. Each agent is provided with an ad-hoc selection of 5-shot task examples from the dataset and receives detailed, context-rich instructions from the Orchestrator—including orchestrator reasoning traces and relevant input data (Wei et al., 2022). The *Content Ordering* agent determines the optimal, fact-preserving sequence for the input data. The *Text Structuring* agent segments this sequence into paragraphs and sentences using explicit tags such as `<paragraph>` and `<snt>`. The *Surface Realization* agent then generates fluent, factually correct text from the structured content. To ensure efficiency and prevent infinite loops, each subtask is limited to three iterations. This modular, iterative approach supports traceability and facilitates targeted regeneration (Xi et al., 2025).

iii. Guardrails: The output of each Worker Agent is rigorously assessed by dedicated, LLM-powered Guardrail Agents, each operating according to structured evaluation criteria expressed in natural-language prompts. Specifically, the *Content Ordering Guardrail* checks for completeness, logical sequencing, and format adherence; the *Text Structuring Guardrail* ensures sentence and paragraph coherence; and the *Surface Realization Guardrails* perform parallel assessments of quality (fluency and grammaticality), flow (coherence and naturalness), and factuality (faithfulness and semantic adequacy). See Appendix 5 for the corresponding prompt templates. Outputs failing to meet requirements are returned with detailed feedback for iterative revision, ensuring that the final output text is both accurate and linguistically appropriate. We adopt a parameter-free optimization strategy called **Prompt-Based Feedback Refinement**, inspired by Reflexion (Shinn et al., 2023) and Self-Refine (Madaan et al., 2023), where guardrail feedback is reformulated into natural-language prompts for subsequent iterations. Each guardrail agent is provided with the Orchestrator's latest instruction, as well as the worker's input and output, enabling precise and actionable evaluation. Whenever feedback other than `CORRECT` is received, it is incorporated into the next prompt—provided the same worker is invoked again — thus enabling the system to iteratively refine its outputs without altering model parameter. For instance, if the surface realization step is flagged for coherence issues, the Orchestrator will instruct the corresponding Worker to improve logical flow in the next round. This dynamic feedback loop allows agents to continuously adapt their strategies, recover from errors, and improve robustness in line with recent trends in prompt-level self-evolution (Du et al., 2025).

iv. Finalizer: After all three Worker Agents have produced validated outputs, the Finalizer uses the outputs and interaction histories from the last two Worker stages to produce the final text. Depending on the results, it may remove structural tags, polish the phrasing, or return the exact text generated by surface realization. This final step, analogous to the summarization modules in systems like ChatEval (Chan et al., 2024), ensures consistency, fluency, and high overall textual quality.

3.2 End-to-End (E2E) System

For comparison, we also implemented an end-to-end (E2E) system that only utilizes an ad-hoc selection of 5-shot example prompting with the GPT-4.1 (OpenAI, 2025a) model. In this setting, a single prompt is used to generate the final output directly from the input data, without modular decomposition, agent specialization, or intermediate guardrail feedback. This E2E approach serves as a baseline to evaluate the effectiveness and interpretability of our proposed multi-agent architecture.

4 Results

4.1 Automatic Metric Results

Table 1 reports the automatic evaluation scores for three systems: our End-to-End (E2E) baseline, the proposed D2T-Agent (GPT-4.1), and StructGPT (Osuji et al., 2024a), which was among the top performers in the GEM-2024 shared task (Mille et al., 2024b). StructGPT adopts a 3-stage pipeline neural architecture, leveraging Flan-T5 (Chung

	BLEU ↑	METEOR ↑	ChrF++ ↑	TER ↓	BERT_F1 ↑	BLEURT ↑	COMET ↑
StructGPT4	49.80	0.40	0.66	0.45	**0.96**	0.56	0.82
E2E	**50.63**	**0.42**	**0.70**	**0.44**	**0.96**	**0.59**	**0.83**
D2T-Agent	48.42	**0.42**	0.69	0.46	**0.96**	0.58	**0.83**

Table 1: Automatic metrics results. Bold and underlined results denote the best and the second best ones respectively.

	Fluency		Grammaticality		No Addition		No Omission	
	LLM	Human	LLM	Human	LLM	Human	LLM	Human
Human	6.35	6.75^{B}	6.52	6.91^{B}	6.63	6.94^{B}	6.70	6.97^{A}
Struct	6.77	6.95^{A}	6.94	6.98^{A}	6.86	$\mathbf{6.98}^{A}$	6.08	6.39^{B}
E2E	6.88	6.94^{A}	6.98	$\mathbf{6.99}^{A}$	**6.92**	$\mathbf{6.98}^{A}$	6.95	6.97^{A}
D2T-Agent	**6.89**	$\mathbf{6.96}^{A}$	**6.99**	6.98^{A}	6.90	6.97^{AB}	**6.96**	$\mathbf{6.98}^{A}$

Table 2: Average scores on a 1–7 Likert scores for LLM-as-Judge and Human Evaluation. Superscript letters mark groups that do not differ significantly in pair-wise Mann–Whitney U tests ($p < 0.05$). Bold and underlined results denote the best and the second best ones respectively.

et al., 2022) for fine-tuning and generation in the CO and TS stages, and GPT-4 (Achiam et al., 2023) for surface realization.

Across most metrics recorded, including BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), ChrF++ (Popović, 2017), TER (Snover et al., 2006), BERT_F1 (Zhang et al., 2020), BLEURT (Sellam et al., 2020), and COMET (Rei et al., 2020), the E2E system consistently achieves the highest scores. The D2T-Agent also demonstrates competitive performance, frequently ranking second across most metrics. StructGPT remains robust, outperforming the multi-agent D2T-Agent in BLEU and TER, while scoring slightly lower in the other evaluation measures.

4.2 Human Evaluation

Three co-authors served as annotators for the human evaluation. The assessment was conducted in a blind fashion, with annotators unaware of the source system for each output. To ensure a representative sample, only examples containing 2 to 7 input triples were included, covering all domains; examples with a single triple were excluded from the evaluation. The evaluation began with a pilot phase in which 20 samples were rated—five outputs each from the three systems and from the human reference. For the main evaluation, annotators assessed a total of 400 samples: 100 outputs from each system and 100 human references drawn from the test set.

Outputs were evaluated on a 7-point Likert scale (1 - lowest, 7 - highest) using four criteria (as shown in Table 2 and Table 3): fluency, grammaticality, no-omission, and no-addition, following the human assessment protocol established in (Mille et al., 2024b). See Appendix 5 for definitions of the evaluation criteria.

In addition to human annotation, we also applied an LLM-as-judge approach, using GPT o3 (OpenAI, 2025b), Claude Sonnet 3.7 by Anthropic (Anthropic, 2024), and Gemini 2.5 Pro (Deepmind, 2025) to assess outputs according to the same criteria, as described in (Mille et al., 2024b). This combination of human and LLM-based judgments provides a comprehensive view of system performance.

4.3 Analysis

As shown in Table 2, the multi-agent approach did not show a significant advantage over the end-to-end GPT-4.1 system; however, LLM-as-a-judge evaluations suggest it may be less prone to content omission.

We attribute the lack of difference between agent-based and end-to-end approaches to the relative simplicity of the dataset and task, which may not sufficiently highlight the benefits of multi-agent coordination and inspection mechanisms. Notably, results indicate that the multi-agent system is competitive with, and in some respects outperforms, the StructGPT pipeline, suggesting that agent-based neural architectures remain a promising direction. The multi-agent framework offers modularity, transparency, and some improvements in omission and fluency, but only marginal gains over the E2E system in the current setup.

Interestingly, some LLM-generated outputs were rated higher than human-written references. However, these findings are based on a relatively simple benchmark and should be considered preliminary, as this work is still ongoing.

It is important to note that, due to the long public availability of the WebNLG dataset, data leakage or

contamination may have occurred—that is, LLMs may have been trained on data closely resembling or identical to the test samples—potentially undermining the validity of these evaluations (Balloccu et al., 2024; Zhou et al., 2023). While large models such as GPT-4.1 may have benefited from this exposure, enabling end-to-end systems to reproduce test set content more easily, our multi-agent approach requires the generation of intermediate structured outputs (content ordering, text structuring), making the process more reliant on reasoning and less on memorization or retrieval.

5 Conclusion

This study presented a new agentic architecture for NLG that combines the structural benefits of classical pipeline architectures with the flexibility of end-to-end models. The new approach was comprehensively assessed against E2E and 3-stage pipeline-based D2T generation systems on a single (albeit simple) benchmark dataset. The results show that E2E achieves the highest scores on most automatic metrics, while the D2T-Agent performs comparably to E2E in both human and LLM-as-judge evaluations.

For future work, we plan to perform evaluations on higher-quality datasets composed of semantic instances with more triples and longer output texts, and to apply data perturbations such as creating fictional entities with LLMs and Counterfactual entities through displacement (Axelsson and Skantze, 2023; Mille et al., 2024b) to assess generalization and minimize contamination risk. We also aim to explore new architectural variations, including single-agent (end-to-end) systems with integrated feedback and iterative refinement, to strengthen baseline comparisons. Additionally, we will experiment with a simplified setup in which a single agent performs all 3-stage pipeline tasks, enabling a deeper analysis of trade-offs between architectural simplicity and control.

Limitations

This study is subject to several important limitations. First, all experiments were conducted on a single, widely used dataset (WebNLG), which poses risks of data contamination, as large language models may have been exposed to the test data during pretraining. As a result, the high performance of end-to-end systems may reflect memorization or retrieval rather than genuine generalization, potentially undermining the validity of the comparison. Second, WebNLG is relatively simple in terms of both the number of facts per instance and the complexity of the required text, which may not fully showcase the advantages of multi-agent coordination and iterative inspection. Third, the human evaluation relied on a small pool of annotators, all of whom were co-authors, which may introduce bias despite efforts to ensure blind and randomized assessment. Finally, the multi-agent framework incurs additional computational cost compared to simpler approaches.

To address some of these limitations, future work will evaluate the framework on larger, more complex, and less publicly available datasets, incorporate data perturbations to better probe generalization, and extend the scale and diversity of human evaluation.

Ethics Statement

All experiments used publicly available data and models. Human evaluation was carried out by co-authors using clear guidelines. No private or sensitive data was used. We acknowledge possible risks related to bias and environmental impact from large language models and encourage ethical use in future research.

Acknowledgements

This work was supported by Research Ireland Centre for Research Training in Artificial Intelligence (CRT-AI) under Grant No. 18/CRT/6223, and by Research Ireland Centre for AI-Driven Digital Content Technology (ADAPT) at Dublin City University under Grant No. 13/RC/2106_P2. We would also like to thank Simon Mille for his helpful assistance in providing the GEM'24 D2T Shared task LLM-as-Judge evaluation script.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2024. Claude. Accessed: 2025-07-06.

Agnes Axelsson and Gabriel Skantze. 2023. Using large language models for zero-shot natural language generation from knowledge graphs. In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG*

Challenge (MM-NLG 2023), pages 39–54, Prague, Czech Republic. Association for Computational Linguistics.

Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondrej Dusek. 2024. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source LLMs. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 67–93, St. Julian's, Malta. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

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+)*, pages 55–76, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. Chateval: Towards better LLM-based evaluators through multi-agent debate. In *The Twelfth International Conference on Learning Representations*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Rossana Cunha, Osuji Chinonso, João Campos, Brian Timoney, Brian Davis, Fabio Cozman, Adriana Pagano, and Thiago Castro Ferreira. 2024. Imaginary numbers! evaluating numerical referring expressions by neural end-to-end surface realization systems. In *Proceedings of the Fifth Workshop on Insights from Negative Results in NLP*, pages 73–81.

Google Deepmind. 2025. Gemini pro - google deepmind.

Shangheng Du, Jiabao Zhao, Jinxin Shi, Zhentao Xie, Xin Jiang, Yanhong Bai, and Liang He. 2025. A survey on the optimization of large language model-based agents. *arXiv preprint arXiv:2503.12434*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, pages 11733–11763.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pages 552–562.

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.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: a survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 8048–8057.

Peyman Heidari, Arash Einolghozati, Shashank Jain, Soumya Batra, Lee Callender, Ankit Arun, Shawn Mei, Sonal Gupta, Pinar Donmez, Vikas Bhardwaj, Anuj Kumar, and Michael White. 2021. Getting to Production with Few-shot Natural Language Generation Models. *SIGDIAL 2021 - 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue, Proceedings of the Conference*, 2:66–76.

Guy Lapalme. 2024. pyrealb at the GEM'24 data-to-text task: Symbolic English text generation from RDF triples. In *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*, pages 54–58, Tokyo, Japan. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Simon Mille, Roberto Carlini, Alicia Burga, and Leo Wanner. 2017. FORGe at SemEval-2017 task 9: Deep sentence generation based on a sequence of graph transducers. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 920–923, Vancouver, Canada. Association for Computational Linguistics.

Simon Mille, Francois Lareau, Stamatia Dasiopoulou, and Anya Belz. 2023. Mod-D2T: A multi-layer dataset for modular data-to-text generation. In *Proceedings of the 16th International Natural Language Generation Conference*, pages 455–466, Prague, Czechia. Association for Computational Linguistics.

Simon Mille, Mohammed Sabry, and Anja Belz. 2024a. Dcu-nlg-small at the gem'24 data-to-text task: Rule-based generation and post-processing with t5-base. In *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*, pages 84–91.

Simon Mille, João Sedoc, Yixin Liu, Elizabeth Clark, Agnes Johanna Axelsson, Miruna Adriana Clinciu, Yufang Hou, Saad Mahamood, Ishmael Nyunya Obonyo, and Lining Zhang. 2024b. The 2024 GEM shared task on multilingual data-to-text generation and summarization: Overview and preliminary results. In *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*, pages 17–38, Tokyo, Japan. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

OpenAI. 2025a. Introducing gpt-4.1 in the api | openai.

OpenAI. 2025b. Introducing openai o3 and o4-mini | openai.

Chinonso Cynthia Osuji, Rudali Huidrom, Kolawole John Adebayo, Thiago Castro Ferreira, and Brian Davis. 2024a. Dcu-adapt-modpb at the gem'24 data-to-text generation task: Model hybridisation for pipeline data-to-text natural language generation. In *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*, pages 66–75.

Chinonso Cynthia Osuji, Brian Timoney, Thiago Castro Ferreira, and Brian Davis. 2024b. Pipeline neural data-to-text with large language models. In *Proceedings of the 17th International Natural Language Generation Conference*, pages 320–329.

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.

Martin J. Pickering and Simon Garrod. 2013. An integrated theory of language production and comprehension. *Behavioral and Brain Sciences*, 36(4):329–347.

Maja Popović. 2017. chrf++: words helping character n-grams. In *Proceedings of the second conference on machine translation*, pages 612–618.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. 2023. Nemo guardrails: A toolkit for controllable and safe llm applications with programmable rails. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 431–445.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *Proceedings of ACL*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Tianyi Zhang, Varsha Kishore*, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Kun Zhou, Yutao Zhu, Zhipeng Chen, Wentong Chen, Wayne Xin Zhao, Xu Chen, Yankai Lin, Ji-Rong Wen, and Jiawei Han. 2023. Don't make your llm an evaluation benchmark cheater. *arXiv preprint arXiv:2311.01964*.

Appendix

Criterion Name	Definition
Fluency	Measures how smoothly and naturally the text reads, including flow, readability, and whether it sounds like language produced by a native speaker.
Grammaticality	Assesses correctness of grammar, syntax, punctuation, and sentence structure throughout the text.
No Omission	Ensures all relevant facts from the input data are included in the generated text; no important information is missing.
No Addition	Verifies that the output does not contain extra or hallucinated information not present in the input triples.

Table 3: Definitions of evaluation criteria for Human Evaluation.

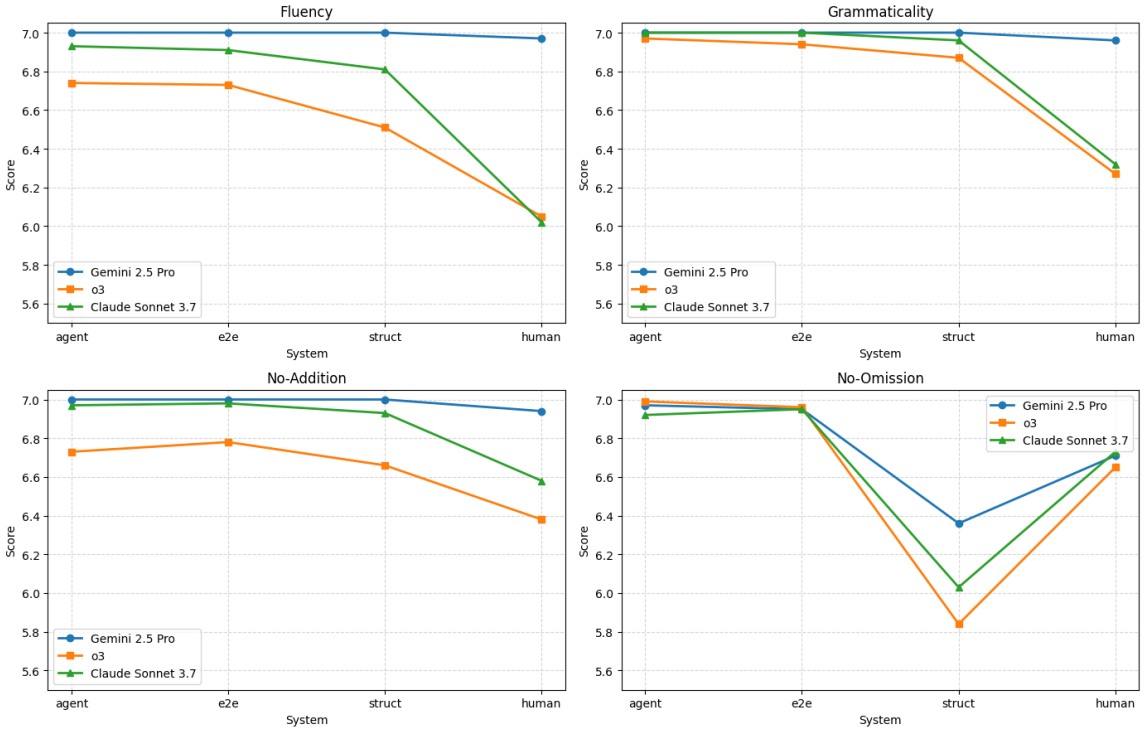

Figure 2: A graph of LLMs Evaluation Scores

Guardrail Agent	Prompt (Instruction)
Orchestrator Prompt	**You are the orchestrator agent responsible for supervising a structured data-to-text generation pipeline.** Your primary role is to ensure the pipeline produces fluent, coherent, and contextually accurate textual outputs that fully align with user expectations. The pipeline comprises three sequential and strictly ordered stages: • **Content Ordering (CO):** Organizes the data logically to form a coherent narrative structure. • **Text Structuring (TS):** Develops organized textual structures such as paragraphs or lists based on ordered content. • **Surface Realization (SR):** Produces the final fluent, grammatically correct, and readable text based on structured content. **WORKFLOW POLICY (Detailed Guidelines)** • **Strict Stage Order:** Always follow the sequence: Content Ordering → Text Structuring → Surface Realization. Do not skip or change the order of these steps under any circumstances. • **Worker Selection:** Assign tasks only to the following named workers: 'content ordering', 'text structuring', 'surface realization', or, for completion, 'FINISH' or 'finalizer'. • **Handling Guardrail Feedback:** If automated guardrail feedback (for accuracy, completeness, or fluency) finds issues, immediately reassign the task to the same worker. Your new instructions must directly address the feedback provided. • **Advance Only on Validation:** Progress to the next stage only after guardrail feedback confirms that the current output is correct, complete, and fluent. • **Improving Surface Realization:** If the surface realization output fails fluency, coherence, or readability checks, reassign the task with explicit guidance for improving naturalness, clarity, and overall quality. • **No Backtracking:** Once a stage is complete and you have moved to the next worker, do not return to previous stages—even if new issues are found later. • **Retry Limit:** If a worker is reassigned the same task three times in a row without producing a satisfactory result, advance to the next stage. • **Avoid Unnecessary Reassignments:** Do not repeat assignments once guardrail feedback confirms all requirements are met, unless there are clearly identified incomplete subtasks. • **Mandatory Feedback Integration:** If the guardrail's OVERALL feedback is 'Rerun `worker` with feedback', reassign the task to that worker and ensure the feedback is included in your new instructions. **WORKER ASSIGNMENT CRITERIA** • Assign clearly named workers based strictly on pipeline progression and outstanding work requirements. • Immediately indicate completion ('FINISH' or 'finalizer') if the full task is successfully completed or if the provided input is insufficient or malformed. • After receiving guardrail feedback labeled 'CORRECT', proceed promptly to the next relevant worker. • If guardrails provide feedback indicating errors, explicitly reassess and revise worker instructions to address the specific errors noted, justifying each reassignment decision clearly within your Thought section. **WORKER INPUT REQUIREMENTS** • Consistently provide every worker with: – The full, original input data provided by the user. – Complete history of prior pipeline results and evaluations. – Explicitly incorporate guardrail feedback into any repeated task assignment, clearly highlighting areas needing improvement. – Clearly state expectations, requirements, and outcomes desired from the worker's efforts. – Strictly prohibit invention of new workers, data fields, or tasks outside the predefined scope. – Incorporate your explicit instructions clearly into your Thought reasoning. **OUTPUT FORMAT** • **Thought:** (Provide a detailed reasoning process based on user requirements, completed stages, guardrail feedback, and clearly justify any task assignments or reassignments.) • **Worker:** (Choose explicitly from: 'content ordering', 'text structuring', 'surface realization', 'FINISH', or 'finalizer'.) • **Worker Input:** (For 'FINISH' or 'finalizer', return the refined final text. For other workers, provide clear, detailed instructions, all relevant data, context, guardrail feedback, and set expectations for the task.) • **Instruction:** (List/outline the task, expectations and supply any specific instructions or tips that will help the worker perform it accurately and efficiently.) **Only include the fields `Thought:`, `Worker:`, `Worker Input:`, and `Instruction:` in your output.**
Worker Prompt	You are a specialized agent responsible for one of three roles: content ordering, text structuring, or surface realization in a data-to-text pipeline. **Your Task:** Carefully complete the task specified in `Worker:` using only the information in `Worker Input:`. Do not add facts that are not present or omit any essential information. **Output Requirements:** • Explain your reasoning clearly and step by step. • Ensure your output is fluent, relevant, and directly based on the input data. • Only include information supported by the data—never hallucinate or invent. • Stay strictly within your assigned role and do not include unrelated content. Focus on accuracy, completeness, and natural language fluency to maximize the quality of your output.
Content Ordering Prompt	You are the content ordering agent in a data-to-text pipeline. **Task Overview** • Arrange structured data in a sequence that best supports the user in generating fluent, coherent, and accurate text. • The goal is to make the order as natural and easy to read as possible, with related facts appearing close together. **Ordering Principles** • Place pieces of information that are logically or thematically related next to each other in the sequence. • Arrange facts to avoid abrupt jumps between unrelated topics, making the information flow smoothly and clearly. • Do not omit, invent, or alter any input information; every input fact must be included exactly as provided. **Terms and Conditions** • **Ordering:** Select the best sequence so that related facts are adjacent or near each other, making it easier for the user to write smooth and cohesive text. • **Related information:** Facts that refer to the same entity, event, or theme, or that build upon each other in a logical or meaningful way. **Examples** • {5 Shot examples} Use your judgment to choose the most logical and human-like ordering, keeping related facts together and enabling clear, coherent, and factually faithful text for the user.

Guardrail Agent	Prompt (Instruction)
Content Ordering Guardrail	You are a guardrail evaluating the output of the 'content ordering' agent in a WebNLG-style data-to-text generation pipeline. **Task:** Determine whether the agent has reordered the extracted triples from the input Triple Set in a way that supports natural, fluent, and logical text generation. **Evaluation Criteria:** • **No-Omissions**: Every fact (triple) from the original input must be present in the output ordering. • **No-Additions**: No new facts, hallucinations, or fabricated information should be present. • **Order**: The sequence should enhance clarity and readability for sentence/paragraph generation, but there is *no single correct order*; accept multiple plausible groupings or sequences. • **Diversity in Style**: Do not penalize alternative, logically sound orderings or grouping styles. Accept nearly correct or reasonable results. • **Strictness**: Flag only if there are true structural issues (illogical jumps, misplaced groupings, clear confusion, or missing/added facts). **How to Judge:** 1. Check all triples are present, no more, no less. 2. Assess if the ordering is reasonable for conversion into coherent sentences/paragraphs. 3. Do not enforce a specific ordering unless required for clarity. 4. Accept unchanged orders if still coherent. **Output Format:** • If all triples are present, and the order is reasonable: respond with **CORRECT** • Otherwise: provide a short, clear explanation (e.g., "Omitted a triple", "Order creates confusion", "Fact hallucinated"). FEEDBACK:
Text Structuring Prompt	You are the text structuring agent in a data-to-text pipeline. **Task Overview** • Group a list of ordered facts into coherent sentences and paragraphs, mirroring how a skilled human writer would present them. • Use <snt> tags for sentences and <paragraph> tags for paragraphs. **Guidelines** • Combine related facts into sentences so the text feels natural and informative. • Group sentences discussing similar topics or entities into the same paragraph. • Avoid creating choppy text with one fact per sentence; include two or more related facts in each <snt> whenever possible. • Do not change, remove, or invent any information—preserve the original sequence and format. • Only add <snt> and <paragraph> tags for structure. The content of each fact must remain unchanged. **Strategy** • Use your judgment to determine which facts belong together in a sentence (<snt>), typically those describing the same entity, event, or theme. • Organize sentences that share a common subject or logical flow into the same paragraph (<paragraph>). • For short lists: use a single paragraph with a few sentences. • For longer lists: divide the text into multiple paragraphs, each with several sentences. **Terms** • **Sentence (<snt>):** A set of facts that naturally belong together and would be expressed in a single sentence by a human writer. • **Paragraph (<paragraph>):** A group of sentences covering related topics, forming a natural and readable unit. **Examples** • {5 Shot examples}
Text Structuring Guardrail	You are a guardrail for the 'text structuring' phase in a WebNLG triple-based data-to-text pipeline. **Task:** Decide if the agent grouped the ordered triples into sensible sentence-level (<snt>) and paragraph-level (<paragraph>) units. **Evaluation Criteria:** • **No-Omissions**: Every triple from the input must be present in the output, grouped into some <snt> (sentence) and <paragraph> (paragraph). • **No-Additions**: No new or hallucinated facts or tags should be introduced. • **Accurate Grouping**: <snt> tags must group related facts for a sentence; <paragraph> tags group related sentences. • **Order Preservation**: The order should follow the content ordering phase, unless there's a strong structural reason. • **Well-Formed Structure**: All tags must be valid and closed. • **Flexibility**: Allow for different—but reasonable—grouping styles. **How to Judge:** • Confirm all triples are included and properly grouped. • Flag only for missing facts, hallucinated content, or broken grouping/structure. **Output Format:** • If the grouping is logical, complete, and no facts are omitted or added: respond with **CORRECT** • Otherwise: give a concise explanation of what is missing or incorrect. FEEDBACK:
Surface Realization Prompt	You are a data-to-text generation agent. Your task is to convert structured content, marked with <snt> and <paragraph> tags, into fluent, coherent, and accurate natural language text. **Goal** • Produce text that fully conveys every fact from the input in clear, well-formed sentences and paragraphs. • The result must be natural and easy to read, with no information added, omitted, or altered. **Instructions** • Convert all input facts into smooth, logically connected natural language. • Do not include any tags, labels, or formatting markers in your output. • Do not invent, omit, or modify any information from the input. • Combine facts from each <snt> block into fluent sentences, but feel free to merge information from multiple <snt> blocks to create richer, more informative sentences when appropriate. • Vary your sentence structure to avoid repetitive or formulaic language. • Make sure to use correct referring expressions (such as proper names, nouns, pronouns, noun phrases, dates and times, titles, numeric or unique identifiers) and determiners. • Use natural paragraphing when the input covers different topics or entities. • Avoid bullet points, lists, or any structured formatting in your output. • Ensure the final text is fluent, grammatically correct, semantically faithful, and easy to read. • Avoid repeating any fact—ensure each piece of information appears only once. • Present the text in a style that is natural, human-like, fluent, clear, and easy to read. **Examples** • {5 Shot examples}

Guardrail Agent	Prompt (Instruction)
Surface Realization (Fluency & Grammaticality)	You are a guardrail focused on evaluating the **fluency** and **grammatical correctness** of a generated text in a data-to-text generation pipeline. You will receive a complete paragraph level or sentence level generated text for evaluation. **Definitions:** • **Fluency**: How smoothly and naturally the output reads. A fluent sentence has appropriate word choice, sentence rhythm, and no awkward or choppy phrasing. • **Grammaticality**: Correctness according to standard grammar rules, including subject-verb agreement, tense consistency, punctuation, and syntactic structure. **Task:** Determine whether the generated output is readable, well-formed, and free of grammatical issues. **Evaluation Criteria:** • **Fluency**: Sentences should read naturally and avoid awkward constructions or unnatural collocations. • **Grammaticality**: The text must be grammatically correct according to formal written English norms. • Penalize the text if there are repetitions such as in facts. **Output Format:** • If both criteria are met: respond with **CORRECT** • If either is violated: return a concise one-sentence specific explanation. FEEDBACK:
Surface Realization (Faithfulness & Adequacy)	You are a guardrail focused on evaluating **faithfulness** to the input data and the **adequacy** of the output content in a data-to-text generation task. You will receive a complete paragraph level or sentence level generated text for evaluation. **Definitions:** • **Faithfulness**: The output must remain factually accurate and reflect only the information present in the input. No fabricated, altered, or hallucinated information is allowed. • **Adequacy**: The output must include all the critical and salient facts from the input data. It should not omit important content necessary for understanding the data. **Task:** Verify that the output is strictly derived from the input and comprehensively conveys its key information. **Evaluation Criteria:** • **Faithfulness**: Every statement in the output must be traceable to the input data. • **Adequacy**: All major data points should be present; the text should not skip or ignore essential facts. **Output Format:** • If both criteria are satisfied: respond with **CORRECT** • If either is violated: return a concise one-sentence specific explanation. FEEDBACK:
Surface Realization (Coherence & Naturalness)	You are a guardrail evaluating whether the generated text is **coherent** and **natural** in a data-to-text generation task. You will receive a complete paragraph level or sentence level generated text for evaluation. **Definitions:** • **Coherence**: How well the ideas and facts in the text are organized and connected. A coherent output has a logical structure and clear flow, even when multiple data points are presented. • **Naturalness**: Whether the output sounds like it was written by a human. It should avoid stilted, robotic, or overly templated language. **Task:** Assess whether the text presents the information in a clear, logically connected manner and reads as if authored by a human. **Evaluation Criteria:** • **Coherence**: Sentences should connect well; transitions between ideas must make sense. • **Naturalness**: The phrasing should resemble that of human writing, not mechanical output. **Output Format:** • If both criteria are met: respond with **CORRECT** • If either is violated: return a concise one-sentence specific explanation. FEEDBACK:
Finalizer Prompt	You are the final agent responsible for generating the final output text based on the results of the data-to-text pipeline. The final output should be fluent, coherent and factually accurate, reflecting the structured data processed through the previous stages. **Your Role** • You are tasked with proofreading, refining and presenting a perfect final text generated by the previous stage. • Extract and return the final natural language text strictly from the 'surface realization' stage if it is verbalised perfectly. • Do not generate, rephrase, or embellish any part of the content. • Ensure the output reflects the final prediction as close as possible to the ground truth. **Instructions** • Only return the surface realization output if it is factually accurate and complete. • The output should match the style, phrasing, and informational structure of the ground truth. • Do not invent details, add stylistic wrappers, or include filler commentary. • If the surface realization result is missing, incomplete, or incorrect, report exactly what is missing. • Remove symbols, tags and special characters (e.g., xml tags `<snt>`, `</snt>`) only keep if they are not necessary. **Output Format** • **Final Answer:** [One fluent, compact sentence that accurately reflects the structured data without deviation]
End-to-End Generation Prompt	You are a data-to-text generation agent. Your task is to generate fluent, coherent, and factually accurate text from structured data. **Objective** • Convert structured input into clear and natural language text that fully and faithfully represents all provided information. Ensure the output is easy to read, highly fluent, and logically connected. **Input Format** • Structured data may be presented as triples, attribute-value pairs, tables, or other standardized formats. **Output Requirements** • Include all information present in the input; do not omit or add facts • Express content using clear, coherent, and well-formed sentences • Prioritize fluency and logical flow throughout the text • Do **not** copy format markers or tags from the input • Do **not** fabricate, infer, or hallucinate information not present in the input • Avoid repetitive or mechanical sentence patterns **Writing Guidelines** • Present information in a logical and connected manner • Use varied and natural sentence structures for better readability • Maintain strict fidelity to the input: no additions, no omissions • Ensure the output is easy to understand and free from awkward phrasing

Guardrail Agent	Prompt (Instruction)
Input Prompt	You are an agent designed to generate text from data for a data-to-text natural language generation. You can be provided data in the form of xml, table, meaning representations, graphs etc. Your task is to generate the appropriate text given the data information without omitting any field or adding extra information in essence called hallucination. **Dataset:** `{dataset_name}` Here is the data, now generate text using the provided data: **Data:** `{data}` **Output:**

Table 4: Prompts for D2T Tasks