Assessing Semantic Consistency in Data-to-Text Generation: A Meta-Evaluation of Textual, Semantic and Model-Based Metrics

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

Abstract

Ensuring semantic consistency between semantic-triple inputs and generated text is crucial in data-to-text generation, but continues to pose challenges both during generation and in evaluation. In order to assess how accurately semantic consistency can currently be assessed, we meta-evaluate 29 different evaluation methods in terms of their ability to predict human semantic-consistency ratings. The evaluation methods include embeddings-based, overlap-based, and edit-distance metrics, as well as learned regressors and a prompted 'LLM-as-judge' protocol. We meta-evaluate on two datasets: the WebNLG 2017 human evaluation dataset, and a newly created WebNLG-style dataset that none of the methods can have seen during training. We find that none of the traditional textual similarity metrics or the pre-Transformer model-based metrics are suitable for the task of semantic consistency assessment. LLM-based methods perform well on the whole, but best correlations with human judgments still lag behind those seen in other text generation tasks.

1 Introduction

The last few years have seen substantial advances in the quality of automatically generated text (Hurst et al., 2024; Dubey et al., 2024) in particular with respect to suprasentential fluency and coherence, thanks to pretrained language models with ever larger numbers of parameters, trained on ever larger datasets (Hoffmann et al., 2022). However, these advances have come at the price of semantic controllability, with confabulation, replacement and omission of content all commonly found in state-of-the-art text generation systems (Hao et al., 2025).

In controlled text generation, whether in the form of data-to-text generation, or of free text generation with given control attributes, some or all of the input must be matched in specific ways by the output. Ensuring this *semantic consistency* between input and output is a known weakness of otherwise state-of-the-art neural generators, and we currently do not have sufficiently reliable methods either (i) for ensuring semantic consistency as part of the neural generation process, or (ii) for evaluating systems in terms of the degree to which they achieve it.

This poses particular problems for tasks like data-to-text generation (Corbelle et al., 2022), where the information conveyed by the output is intended to be entirely controlled by the information contained in the input. Currently the only way to perform such *semantic consistency assessment* reliably is manual evaluation, with supervision-trained automatic methods getting up to about three quarters of unseen assessments right (Dušek and Kasner, 2020; Liu et al., 2023; Cui et al., 2024). An automatic method that can reliably assess the semantic consistency between inputs and outputs directly would be useful not only for evaluation in and post development, but also potentially as part of the text generation method itself, e.g. via reranking of outputs (Harkous et al., 2020), or as part of a loss function. However, semantic consistency continues to be challenging to assess automatically with a high degree of reliability, and evaluation by comparison to human-written reference outputs or by human evaluators is predominantly used instead.

In this paper, we evaluate diverse types of semantic consistency evaluation methods on two datasets: the WebNLG 2017 human evaluation dataset (Dataset A), and a newly created, unseen WebNLG-style dataset (Dataset B). Moreover, we test each method under two conditions: (i) with triple inputs 'textified' before assessment, and (ii) without. Our main contributions are:

1. A new WebNLG-style data-to-text dataset of new triple inputs and corresponding texts for people and city entities sampled from the GREC corpus (Belz et al., 2009), with out-

Proceedings of the 18th International Natural Language Generation Conference, pages 98–107
October 29–November 2, 2025.

puts generated by five different NLG systems.

2. Human semantic consistency annotations for a subset of 100 input-output pairs from the above new dataset. These are WebNLG 2017 style annotations for 'semantic adequacy' criterion.
3. An empirical evaluation of 29 evaluation methods for semantic consistency on two data-to-text corpora, one established, one new; and under two conditions: with input triples textified, and without.
4. A novel LLM-as-judge protocol where LLMs (Command-r-plus, Llama3-70B, Mistral-7B) rate semantic consistency on a 3-point scale with for-human instructions, and their scores are ensembled into a single metric.
5. An analysis of how model type, input representation and underlying definition of semantic consistency affect correlation with human ratings.

2 Related Work

Semantic consistency assessment (SCA) is considered challenging (Harkous et al., 2020; Liu et al., 2023; Cui et al., 2024). Existing methods tend to not directly use the structured meaning representations (SMRs) that typically form the input to data-to-text generation, but first (trivially) map them to text, before applying either a semantic similarity measure (Mille et al., 2023), or performing a natural language inference (NLI) task (Dušek and Kasner, 2020), on the (mapped) inputs and corresponding textual outputs.

Such methods tend to not generalise well beyond the data they were trained on. Moreover, pipelining multiple processes aggregates errors. Harkous et al. (2020) presented an end-to-end data-to-text generation system with a semantic fidelity classifier for semantically inaccurate text detection. Faille et al. (2021) proposed an automatic metric for assessing entity-based semantic adequacy of RDF verbalisers. Ribeiro et al. (2020) used natural language inference (NLI) to detect two-way entailment between generated text and the input.

In existing work, semantic consistency assessment has been construed e.g. as binary classification (Harkous et al., 2020), mutual textual entailment (Dušek and Kasner, 2020), or thresholded semantic similarity (Mille et al., 2023), between inputs and outputs (see also related research in Section 2). However, little attention has been paid to underlying definitions of semantic consistency and the role they play in evaluation results which can vary substantially depending on which definition underlies an evaluation method.

3 Datasets

We evaluate two corpora of RDF-text pairs with human judgments of semantic adequacy:

Dataset A (WebNLG 2017): A subset of the public WebNLG 2017 human evaluation data (Shimorina et al., 2018), comprising 100 randomly sampled input-output pairs (out of 2,230 total) across ten systems. Each text was rated by three annotators on a 3-point scale for semantic adequacy ("Does the text correctly represent the input triples?"), and we use the mean score.

Dataset B (GREC-derived): A new WebNLG-style benchmark built from the People (442) and Cities (243) entities in the GREC 2.0 corpus (Belz et al., 2009). We automatically extract RDF triples for 100 sampled entities (see Algorithms 1–2 in Appendix A), generate verbalisations with five available NLG systems previously submitted to the WebNLG shared tasks (FORGe, CycleGT, DCU-NLG-Small, DCU-NLG-PBN, DCU-ADAPT-modPB, see details in Appendix B), and collect three independent, human-assessed semantic-consistency ratings per example following the WebNLG 2017 protocol. Ratings are averaged for our meta-evaluation. We will release all of our data here.[1]

4 Evaluation Methods

We meta-evaluate 29 evaluation methods in terms of their ability to predict human semantic consistency ratings; the methods fall into three broad categories depending on the underlying (implied) definition of semantic consistency:

1. Textual similarity between inputs and outputs (ROUGE, BLEU, edit-distance, and set-overlap metrics): the more similar the texts, the greater the semantic consistency;
2. Semantic similarity between inputs and outputs (embeddings-based metrics): the more similar the number vectors, the greater the semantic consistency; and

[1] https://github.com/mille-s/Build_KGs_entities.git

3. Model-based assessment (learned regressors, LLMs): semantic consistency is as good as the model predicts it to be.

We deliberately include both long-established metrics and more recent model-based methods. While some traditional overlap or distance-based metrics are considered outdated in text-to-text evaluation, they continue to be used in some parts of the research community and therefore remain relevant for comparison. Their inclusion also allows us to quantify more precisely how much newer metrics and LLM-based evaluators improve upon them in the more challenging data-to-text setting, thereby providing a comprehensive baseline for future work.

More specifically, we use the following 29 automatic methods (see Appendix C for details) falling into five method types:

- **Embeddings-based semantic similarity**: BERTScore P/R/F_1; SBERT Cosine, Dot, Euclidean, Manhattan.
- **Overlap-based textual similarity**: SacreBLEU; ROUGE-1/2/L P/R/F1; METEOR.
- **Edit-distance and set-based textual similarity**: Levenshtein; Jaccard similarity, distance; Dice.
- **Assessment by learned regressors**: BLEURT, InferSent, Universal Sentence Encoder.
- **Assessment by LLMs**: Command-r-plus, Llama3-70B, Mistral-7B, and the three models ensembled.

The above methods are used to obtain semantic-consistency assessments on data-to-text input/output pairs in our two datasets, with and without textification of inputs.

5 Experimental Set Up

Our aim is to determine which of the metrics from the preceding section best predict human judgements of semantic consistency.

In data-to-text generation, human evaluation remains the gold standard, but is expensive and time-consuming (Thomson et al., 2024; Sellam et al., 2020). Consequently, automatic proxies are widely used, including **embeddings-based** (BERTScore (Zhang et al., 2019); SBERT (Reimers and Gurevych, 2019a); BLEURT (Sellam et al., 2020)), **overlap** (ROUGE (Lin, 2004); SacreBLEU (Post, 2018); METEOR (Banerjee and Lavie, 2005)), and **edit/set** measures (Levenshtein (Levenshtein et al., 1966); Jaccard/Dice (Jaccard, 1901; Dice, 1945)). We assess all of these in our experiments. However, they were designed for text-to-text tasks, not structured inputs, so we test them both with and without textification.

We extend our tests to LLM-as-judge methods, prompting three pre-trained models (Command-r-plus, Llama3-70B, Mistral-7B) to rate semantic consistency on a 1-3 scale, and averaging over three seeds (42; 1234; 1738).

We evaluate the metrics and LLM judges on the two datasets from Section 3, once with inputs as they are (structured triples), and once with linearised and 'textified' triples (see Algorithm 3 in Appendix A). Our simple textification method replaces special characters with spaces, collapses whitespace, and trims edges.

We compute scores with all evaluation methods, then compute Pearson's r between the scores and mean human ratings for each dataset, ranking methods by correlation to identify the strongest predictors.

6 Result and Analysis

The results in Table 1 reveal systematic differences in metric performance between the two datasets, and between textified and non-textified inputs.

First, textification exerts a positive effect on most of the evaluated metrics on Dataset A (except BLEURT, Rouge-L Recall, Rouge-1 Recall, SacreBLEU, Jaccard Distance), yielding an average Pearson's r increase of about 0.07; and also on Dataset B (here the exceptions are the four SBERT metrics, ROUGE-2 Precision, SacreBLEU, Levenshtein Distance, Jaccard Distance) where the average increase in r is about 0.04.

Textification has a slightly smaller beneficial effect on Dataset B correlations for the LLM judges, except Mistral. In stark contrast, it causes LLM correlations to collapse across the board on Dataset A (see also Discussion section).

Among the metrics, the four SBERT-based similarity metrics with textification attain the four highest correlations on Dataset A ($r = 0.582$), substantially outperforming BLEURT (0.530). In contrast, BLEURT ($r = 0.733$), followed by BERTScore F1 (0.670) and then the SBERT variants (0.655)

Type of Metrics	Metrics	Dataset A (WebNLG 2017)		Dataset B (new WebNLG-style dataset	
		r (-Textification)	r (+Textification)	r (-Textification)	r (+Textification)
Automatic Metrics	SBERT (Euclidean)	0.507	**0.582**	**0.655**	**0.655**
	SBERT (Manhattan)	0.502	**0.582**	**0.662**	**0.655**
	SBERT (Cosine)	**0.518**	**0.573**	0.654	0.638
	SBERT (Dot Product)	**0.518**	0.573	0.654	0.638
	BLEURT	**0.530**	0.501	**0.713**	**0.733**
	Universal Sentence Encoder	0.248	0.500	0.271	0.402
	BERTScore R	0.417	0.447	0.431	0.609
	BERTScore F_1	0.421	0.441	0.572	**0.670**
	BERTScore P	0.424	0.433	0.636	0.647
	InferSent	0.372	0.379	-0.148	-0.022
	METEOR	0.206	0.379	0.296	0.306
	ROUGE-L F_1	0.305	0.338	0.223	0.356
	ROUGE-L Precision	0.287	0.337	0.248	0.304
	ROUGE-1 Precision	0.292	0.330	0.268	0.305
	ROUGE-1 F_1	0.309	0.329	0.246	0.387
	ROUGE-L Recall	0.328	0.324	0.158	0.281
	Dice Coefficient	0.290	0.321	-0.021	0.074
	Jaccard Similarity	0.290	0.320	-0.010	0.078
	ROUGE-1 Recall	0.329	0.313	0.182	0.323
	ROUGE-2 Precision	0.203	0.264	0.111	0.078
	ROUGE-2 F_1	0.202	0.263	0.097	0.127
	ROUGE-2 Recall	0.198	0.257	0.051	0.082
	SacreBLEU	0.228	0.198	0.400	0.127
	Levenshtein Distance	-0.044	-0.062	-0.384	-0.400
	Jaccard Distance	-0.290	-0.320	0.010	-0.078
LLM-as-Judge	Command-r-plus + Llama + Mistral	**0.635**	**0.367**	**0.716**	**0.720**
	Mistral	**0.531**	**0.362**	0.567	0.488
	Llama	**0.569**	**0.284**	**0.644**	**0.689**
	Command-r-plus	0.518	0.214	**0.727**	**0.729**

Table 1: Pearson's r for Datasets A and B, sorted (within Automatic Metrics and LLM-as-Judge) by Dataset A's r (+Textification). Top three Pearson's r in each group bolded.

perform best on Dataset B.

Among the LLM judges, the three-model ensemble (Command-r-plus + Llama + Mistral) correlates best with human judges on Dataset A ($r = 0.635$), whereas on Dataset B, the single model Command-r-plus achieves the highest correlation ($r = 0.727$), marginally surpassing both the ensemble (0.716) and Llama (0.644).

On Dataset A, the LLM judges without textification, and the model-based metrics with textification, perform best overall and more or less on a par. On Dataset B, the standout performances are by BLEURT, the LLM ensemble and Command-r-plus by a considerable margin, and these are hardly affected by textification. Moreover, on Dataset B, the more traditional metrics fail almost entirely (see rows from InferSent down to Jaccard).

7 Discussion

Among the clearest patterns in our results is the uniformly poor performance of traditional textual similarity metrics at the semantic consistency assessment task. InferSent and Universal Sentence Encoder, the two neural but pre-Transformer metrics, can also be dismissed for this task.

Another clear pattern is that while the traditional textual similarity metrics perform very similarly on Dataset A and Dataset B, every evaluation method that involves an LLM sees a jump in correlation from Dataset A to B. It's not entirely clear why the latter should agree more with humans on B than on A. It may in part be due to the fact that outputs in Dataset B are generally of higher quality, perhaps displaying more minor as well as fewer errors.

Textification caused the LLM-judge scores to collapse (while improving almost all other scores) on Dataset A. The latter was almost certainly ingested as part of their training data by the four LLMs tested, so textification had the net effect of obscuring whether an input and output belonged together (hence their semantic relatedness).

It is worth noting that the highest correlations with human judgments in our experiments were just over $r = 0.7$. While such values fall below text-to-text generation tasks where correlations above 0.8 are commonly reported, they nevertheless represent the current performance ceiling for semantic consistency in data-to-text generation. This gap highlights both the greater intrinsic difficulty of the task and the lack of specialised metrics designed for structured input-output matching. The poor performance of older metrics confirms their unsuitability for semantic consistency assessment, and provides a baseline against which to measure progress with

newer methods.

8 Conclusion

We have presented a meta-evaluation of 29 different evaluation methods on the task of assessing semantic consistency between triple inputs and corresponding textual output in data-to-text generation. The evaluation methods came in three broad flavours (reflecting the underlying definition of semantic consistency): textual similarity, semantic similarity, and model assessment. Meta-evaluation on the WebNLG 2017 human evaluation dataset and a newly created WebNLG-like dataset revealed that none of the older metrics, including those based on pre-Transformer neural models, are suitable in the least for the data-to-text semantic consistency assessment task.

At the same time, no method surpassed Pearson's $r = 0.635$ (LLM ensemble) on Dataset A, or $r = 0.733$ (BLEURT) on Dataset B, with the best prompted LLM-as-judge evaluation methods close behind. These results underscore the notable gap in correlations with human assessments between data-to-text and text-to-text evaluation benchmarks; in the latter, correlations well over 0.8 are routinely reported. One route that may be worth exploring in future work is different ensembling strategies combining the best embeddings-based metrics with LLM-as-judge methods. Improvements in SCA methods can in turn lead to improvements in both data-to-text evaluation and generation.

Finally, by evaluating both older and state-of-the-art methods side by side, we have provided a comprehensive picture of the current landscape. While some of the older metrics are known to be less suited for the purpose of SCA, their continued use in the community and their role as comparative baselines make their inclusion informative. Our experiments were restricted to WebNLG-style English datasets, and it will be important to test whether the observed patterns hold across different domains and languages, to establish the generalisability of the findings.

Limitations

Our experiments showed promising correlations among human metrics, automatic metrics, and LLM evaluations. However, because we examined only a limited set of models and traditional automatic metrics, we cannot generalise these findings beyond that scope.

Ethics Statement

As a paper that compares existing human evaluation results with automatic metrics and LLM-evaluation results, the associated risk was minimal.

Acknowledgments

We extend our sincere gratitude to all the anonymous reviewers for the helpful feedback and advice.

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 Research Ireland Centre for Research Training in Digitally-Enhanced Reality (d-real) under Grant No. 18/CRT/6224. Mille's contribution was funded by the European Union under the Marie Skłodowska-Curie grant agreement No 101062572 (M-FleNS), and by the Irish Department of Tourism, Culture, Arts, Gaeltacht, Sport and Media via the eSTÓR project.. Thomson's contribution was funded by the ADAPT SFI Centre for Digital Media Technology. 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

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.

Anya Belz, Eric Kow, Jette Viethen, and Albert Gatt. 2009. Generating referring expressions in context: The grec task evaluation challenges. In *Conference of the European Association for Computational Linguistics*, pages 294–327. Springer.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, and 1 others. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

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.

Wendi Cui, Zhuohang Li, Damien Lopez, Kamalika Das, Bradley A. Malin, Sricharan Kumar, and Jiaxin Zhang. 2024. Divide-conquer-reasoning for consistency evaluation and automatic improvement of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 334–361, Miami, Florida, US. Association for Computational Linguistics.

Lee R Dice. 1945. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Ondřej Dušek and Zdeněk Kasner. 2020. Evaluating semantic accuracy of data-to-text generation with natural language inference. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 131–137.

Juliette Faille, Albert Gatt, and Claire Gardent. 2021. Entity-based semantic adequacy for data-to-text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1530–1540, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. 2020. Cyclegt: Unsupervised graph-to-text and text-to-graph generation via cycle training. *arXiv preprint arXiv:2006.04702*.

Yijie Hao, Haofei Yu, and Jiaxuan You. 2025. Beyond facts: Evaluating intent hallucination in large language models. *arXiv preprint arXiv:2506.06539*.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2410–2424.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Paul Jaccard. 1901. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579.

Vladimir I Levenshtein and 1 others. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Michela Lorandi and Anya Belz. 2024. Dcu-nlg-pbn at the gem'24 data-to-text task: Open-source llm peft-tuning for effective data-to-text generation. In *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*, pages 76–83.

Simon Mille, Stamatia Dasiopoulou, and Leo Wanner. 2019. A portable grammar-based nlg system for verbalization of structured data. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, page 1054–1056, New York, NY, USA. Association for Computing Machinery.

Simon Mille, Josep Ricci, Alexander Shvets, and Anya Belz. 2023. A pipeline for extracting abstract dependency templates for data-to-text natural language generation. *Depling 2023*, page 91.

Simon Mille, Mohammed Sabry, and Anya Belz. 2024. 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.

Chinonso Cynthia Osuji, Rudali Huidrom, Kolawole John Adebayo, Thiago Castro Ferreira, and Brian Davis. 2024. 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.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

Nils Reimers and Iryna Gurevych. 2019a. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019b. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Leonardo F. R. Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020. Modeling global and local node contexts for text generation from knowledge graphs. *Transactions of the Association for Computational Linguistics*, 8:589–604.

David J. Rogers and Taffee T. Tanimoto. 1960. A computer program for classifying plants. *Science*, 132(3434):1115–1118.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696*.

Anastasia Shimorina, Claire Gardent, Shashi Narayan, and Laura Perez-Beltrachini. 2018. WebNLG Challenge: Human Evaluation Results. Technical report, Loria & Inria Grand Est.

Craig Thomson, Ehud Reiter, and Belz Anya. 2024. Common flaws in running human evaluation experiments in nlp. *Computational Linguistics*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

A Algorithms

Algorithm 1: Property-to-Entity Mapping Construction

Input:
(a) `GREC_NE.json`: list of entities by category
(b) Ontology property definitions: set of valid properties for each category from the Ontology
(c) DBpedia triple data: all RDF triples (subject, property, object) from DBpedia
Output:
`dico_entities_for_triple_configuration_GREC_NEs.json`:
property-to-entity mapping

if `dico_entities_for_triple_configuration_GREC_NEs.json` *does NOT exist* **then**
  for *each category in* `GREC_NE.json` **do**
    for *each property relevant to the category (from Ontology)* **do**
      for *each entity in the category* **do**
        if *there exists at least one triple in DBpedia where the entity is subject and the predicate is the property* **then**
          Add the entity to the property's entity list for that category;
  Save as `dico_entities_for_triple_configuration_GREC_NEs.json`;

Algorithms 1 and 2 show how we carried out triple collection for Dataset B. Algorithm 3 present the approach for the "textification" of triples.

The steps we follow in Algorithm 1 are as follows:

- Read the list of named entities grouped by category from `GREC_NE.json`.
- For each category (e.g., *People*, *Cities*):
  - Retrieve the set of valid properties for the category, as defined in the Ontology.
  - For each entity in the category:
    - Check DBpedia for at least one RDF triple where the entity appears as the subject and the property as the predicate.
    - If such a triple exists, add the entity to the list for that property in the current category.
- Repeat this process for all categories, properties, and entities.
- Save the resulting mapping (category → property → [entities]) to `dico_entities_for_triple_configuration_GREC_NEs.json`.
- This mapping enables targeted triple extraction, which we use in the second algorithm.

The steps we follow in Algorithm 2 are as follows:

- Check whether the file `dico_input_contents_DBp_GREC_NEs.pickle` exists.
  - If not:
    - Traverse the property-to-entity mapping from Algorithm 1.
    - For each category, property, and entity:
      - If the entity is in the target list from `GREC_NE.json`, extract all RDF triples from DBpedia where the entity is the subject and the property is the predicate.
      - Aggregate all such triples per entity.
    - Save the resulting entity-to-triples mapping as `dico_input_contents_DBp_GREC_NEs.pickle`.
  - If the pickle file already exists, load it directly.
- Build a list of all target entities and their categories from `GREC_NE.json`.
- For each entity:
  - Look up the set of valid properties using the property-to-entity mapping.

- Retrieve all available triples for the entity from the pickle file.
- Filter the triples to keep only those with valid properties.
- Optionally, remove triples with unsuitable object values.
- Set the target number N of triples per entity according to the WebNLG distribution (e.g., $N = 3$).
- If more than N triples remain, randomly select N; otherwise, keep all.

- Group the final triples for each entity.
- Write the grouped triples to an XML file for downstream use. This is our 'Dataset B.'

B Systems

We collected a total of 100 samples, with each of the following five systems generating 20 samples:

1. FORGe (Mille et al., 2019) is a portable grammar-based system that maps predicate-argument structures onto sentences by applying a series of rule-based graph-transducers.

2. CycleGT (Guo et al., 2020) is a weakly supervised framework that iteratively bootstraps generation and semantic parsing models by mapping between two modalities (unaligned text and RDF data) to enable joint model improvement.

3. DCU-NLG-Small (Mille et al., 2024) is a combination of FORGe rule-based system with a language model of reduced size (T5), where the rule-based system converts input triples into semantically correct English text and then a language model to paraphrase these text to make it more fluent.

4. DCU-NLG-PBN (Lorandi and Belz, 2024) is a Mistral 7B Instruct model fine-tuned with Low-Rank Adaptation (LoRA) to improve performance while maintaining computational efficiency.

5. DCU-ADAPT-modPB (Osuji et al., 2024) explores two approaches: (1) using a fine-tuned Flan-T5-large model for triple ordering and structuring, followed by prompt-based surface realisation with five-shot prompting; and (2) directly generating text from input triples using a prompt-based model with five examples. We use the latter approach with the Mistral 7B Instruct model.

C Metrics used

We define semantic consistency as the requirement that an output's content fully aligns with, and does not exceed, the information in its input. To assess both semantic similarity and dissimilarity under this definition, we use the following 25 automatic metrics:

- **Embedding-based metrics (BERTScore, SBERT variants)**:
  - **BERTScore:**[2] BERTScore (Zhang et al., 2019) compares two sequences by computing similarity between contextual embeddings produced by pretrained Transformer models. It evaluates how well the tokens in one sequence align with those in the other using cosine similarity.
    - **Precision (P)**: Average of the maximum similarity each token in the first sequence has with tokens in the second.
    - **Recall (R)**: Average of the maximum similarity each token in the second sequence has with tokens in the first.
    - $\mathbf{F}_1$: Harmonic mean of Precision and Recall.
  - **SBERT:**[3] Sentence-BERT (Reimers and Gurevych, 2019b) produces dense sentence-level embeddings using a Siamese or triplet network based on BERT. The similarity between sequences is calculated via standard vector-based measures:
    - **Cosine:** $\frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$, in $[-1, 1]$.
    - **Dot Product:** $\mathbf{u} \cdot \mathbf{v}$, unbounded.
    - **Euclidean:** $\|\mathbf{u} - \mathbf{v}\|_2 = \sqrt{\sum_i (u_i - v_i)^2}$.
    - **Manhattan:** $\|\mathbf{u} - \mathbf{v}\|_1 = \sum_i |u_i - v_i|$.

- **Overlap-based metrics (SacreBLEU, ROUGE, METEOR)**:
  - **SacreBLEU:**[4] SacreBLEU (Post, 2018) standardises the BLEU metric by controlling for tokenisation, smoothing, and formatting, allowing consistent comparison. It computes n-gram precision and penalises overly short outputs via a brevity penalty.
  - **ROUGE:**[5] ROUGE (Lin, 2004) measures the overlap between sequences based on n-grams and longest common subsequences (LCS). It emphasises recall, capturing how much of a reference is matched.
    - **ROUGE-1**: Based on unigram overlap.
    - **ROUGE-2**: Based on bigram overlap.
    - **ROUGE-L**: Based on longest common subsequence.
      - Precision: LCS length divided by candidate length.
      - Recall: LCS length divided by reference length.
      - F_1: Harmonic mean.
  - **METEOR:**[6] METEOR (Banerjee and Lavie, 2005) aligns unigrams between sequences using exact, stemmed, and synonym matches. It balances precision and recall, and applies a fragmentation penalty to discourage disordered matches.

- **Edit- and set-based metrics (Levenshtein, Jaccard, Dice)**:
  - **Levenshtein Distance:**[7] Levenshtein (Levenshtein et al., 1966) computes the minimum number of character edits (insertions, deletions, substitutions) needed to transform one string into another. It can be normalised to obtain a similarity score.
  - **Jaccard Similarity:**[8] The Jaccard Index (Jaccard, 1901) measures similarity between two sets as the

[2] https://github.com/Tiiiger/bert_score

[3] https://www.sbert.net

[4] https://github.com/mjpost/sacrebleu

[5] https://pypi.org/project/rouge/

[6] https://huggingface.co/spaces/evaluate-metric/meteor/tree/main

[7] https://pypi.org/project/Levenshtein/

[8] https://www.geeksforgeeks.org/jaccard-similarity/

ratio of their intersection to their union:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

  - **Jaccard Distance:**[9] A dissimilarity measure defined as $1 - J(A, B)$ (Rogers and Tanimoto, 1960).
  - **Dice Coefficient:**[10] The Dice coefficient (Dice, 1945) measures similarity as:

$$D(A, B) = \frac{2|A \cap B|}{|A| + |B|},$$

    which weights shared elements more heavily than Jaccard.

- **Learned regressors (BLEURT, InferSent, USE)**:
  - **BLEURT:**[11] BLEURT (Sellam et al., 2020) is a regression model fine-tuned on perturbed and human-annotated sentence pairs to predict quality scores aligned with human judgement.
  - **InferSent:**[12] InferSent (Conneau et al., 2017) is a sentence embedding model trained on Natural Language Inference datasets. It produces embeddings useful for similarity tasks using cosine or other distance measures.
  - **Universal Sentence Encoder:**[13] USE (Cer et al., 2018) generates fixed-length embeddings from various architectures trained on multitask learning objectives. Similarity is typically computed using cosine distance.

We apply these metrics to our input/output pairs, where the input RDF triple(s) are processed either with or without textification (see Algorithm 3 for further details).

[9] https://en.wikipedia.org/wiki/Jaccard_index

[10] https://en.wikipedia.org/wiki/Dice-SÃ¿rensen_coefficient

[11] https://github.com/google-research/bleurt

[12] https://github.com/facebookresearch/InferSent

[13] https://www.tensorflow.org/hub/tutorials/semantic_similarity_with_tf_hub_universal_encoder

Algorithm 2: Triple Extraction, Filtering, Sampling, and WebNLG-like XML Data Creation

Input:
(a) `GREC_NE.json`
(b) `dico_entities_for_triple_configuration_GREC_NEs.json`
Output: XML file containing the WebNLG-like dataset, grouped by entity

if `dico_input_contents_DBp_GREC_NEs.pickle` *does NOT exist* **then**
  foreach *each category in* `dico_entities_for_triple_configuration_GREC_NEs.json` **do**
    foreach *each property in the category* **do**
      foreach *each entity in the property's entity list* **do**
        if *the entity is in the target list from* `GREC_NE.json` **then**
          Extract all (subject, property, object) triples for that entity from DBpedia data;
          Add these triples to the list of triples for the entity in the mapping;
  Save as `dico_input_contents_DBp_GREC_NEs.pickle` then Load;
else
  Load `dico_input_contents_DBp_GREC_NEs.pickle`;

Initialize an empty list `target_entities`;
foreach *each category in* `GREC_NE.json` **do**
  foreach *each named entity in the category* **do**
    Add a record (entity name, category) to `target_entities`;

foreach *each record in* `target_entities` **do**
  Let `entity` be the entity name and `category` its category;
  Retrieve all properties for which `entity` is listed in `dico_entities_for_triple_configuration_GREC_NEs.json`;
  Save these as the valid properties for `entity`;

foreach *each entity in* `target_entities` **do**
  Retrieve available triples for this entity from the pickle;
  For each (entity, property) pair: Get the actual triple (subject, property, object) from the pickle file;
  Keep only triples whose property is valid for this entity;
  Optionally remove triples with unsuitable object values;
  Set N as the target number of triples for this entity (e.g., $N = 3$);
  if *number of triples* $> N$ **then**
    Randomly select N triples;
  else
    Keep all triples;
  Save these triples for the entity;

foreach *each entity in* `target_entities` **do**
  Group the final triples for the entity;

Write all grouped triples to an XML file;

Algorithm 3: Minimal Normalisation of Linearised Triples

Input:
List of linearised triple strings L
Output:
List of cleaned and normalised triple strings S

$S \leftarrow$ empty list;
for *each* $line$ *in* L **do**
    Replace all occurrences of `|`, `_`, and `<br>` in $line$ with a single whitespace;
    Replace all consecutive whitespace in $line$ with a single whitespace;
    Trim leading and trailing whitespace in $line$;
    Append $line$ to S;
return S;