A Tool for Collecting Validated Triple Sets for Data-to-Text Generation

Mark Andrade♠, Simon Mille♠♣, Anya Belz♠, Brian Davis♠
♠ ADAPT Research Centre, Dublin City University, Ireland
firstname.lastname@adaptcentre.ie,
♣ Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400 Talence, France
simon.mille@u-bordeaux.fr

Abstract

For evaluating all types of data-to-text Natural Language Generation systems, data should be based on inputs with controlled vocabularies and only correct information. We present a tool that allows for collecting new DBpedia triple sets for any entity while validating the collected information with different levels of leniency.

1 Introduction

Standard Natural Language Generation datasets are now likely to have been exposed to models during training, and cannot reliably be used for assessing model performance (Balloccu et al., 2024). To remedy this situation, recent efforts have been made to collect on-demand uncontaminated input data, for example in the evaluation of table-to-text generation, by collecting Wikipedia table contents from pages created after a cut-off date (Onderková et al., 2025). This approach does provide good testing data for Large Language Models: the system is able to collect a large quantity of very varied contents, and since the contents are not checked during the process, the inputs can contain information that is challenging to verbalise (e.g. because it contains wrong values). However, such data will not allow for evaluating other types of lighter systems, such as some rule-based or traditional learning-based approaches that do not operate on open vocabularies or are not robust to ill-formed data.

In this paper, we propose a tool for collecting validated input data from the current set of DBpedia properties and English DBpedia entities, both of which are closed (although evolving) vocabularies.[1] Compiling triple sets is straightforward but DBpedia contains a significant amount of syntactically correct but non-interpretable knowledge, so a control mechanism for the quality of the collected triples is needed. Consider for example the triple *Ibn al-Tilmidh* || *occupation* || *Baghdad*), where the entities *Ibn al-Tilmidh* (**Subject**) and *Baghdad* (**Object**) are a person and a city respectively, but *occupation* (DBpedia **Property**) is meant to express that the Object is the activity of the Subject. If such a triple were paired with a reference sentence or compared to a predicted output, what would be an acceptable verbalisation of it? Below, we describe our method and tool to filter as many problematic triples as possible while avoiding leaving aside good triples. A demonstration video is available, [2] and all the code can be found at https://github.com/andradeM17/DBpedia.

2 Validation and recovery of triples

2.1 Validation strategy

We use DBpedia's knowledge model to validate triples, by comparing the **expected** types of the Subject and Object entities as defined in the DBpedia ontology to the **actual** types of the Subject and Object entities.[3] For instance, the property *birthDate* expects an entity of class *Animal* as Subject, and a entity of class *Date* as Object. When retrieving triples for an entity, we collect both the type(s) of the entity and the type(s) of the other entity in each triple. Each expected/actual type can be a specific type or undefined.[4] We then compare expected and actual types and assign one of the 4 values below to each triple: (i) *Invalid*: An actual type contradicts one expected type; (ii) *Possibly_Valid*$_E$: No contradiction, but an Entity does not have a dbo:type on DBpedia; *Possibly_Valid*$_P$: No contradiction, but a Property does not specify the type of

[1] ∼1.2K properties and ∼6M entities are currently in use.

[2] https://drive.google.com/file/d/1DvWXc73EEau_SbckIKLIcaC-ejKey2n9/view?usp=sharing

[3] Note that the validation only assesses the compatibility of the entity types in each triple against the current DBpedia ontology model; it does not assess, e.g., inconsistencies between triples or the coverage that a triple set has of the knowledge about an entity. More details in (Andrade et al., 2026).

[4] Superclasses are included into the actual types lists.

Proceedings of the 19th International Natural Language Generation Conference: System Demonstrations, pages 8–10
October 17-21, 2026 ©2026 Association for Computational Linguistics

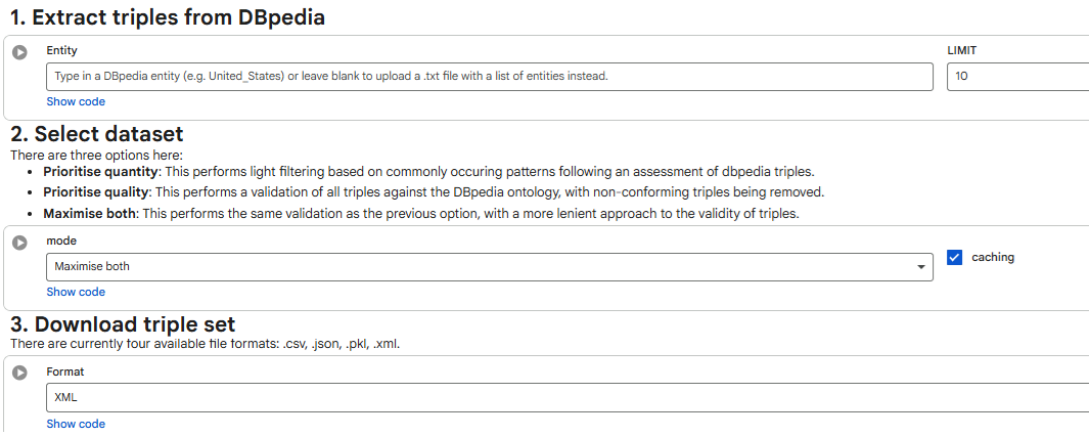


Figure 1: Interface screenshot.

Subject and/or Object it expects; (iv) *Valid*: Actual and expected types match. By filtering through the three non-*Valid* categories, one can create datasets with either more triples (more recall), or less triples of more controlled quality (more precision), as shown on the left part of Figure 2.

2.2 Recovery strategy and evaluation

We applied our validation approach to the entities of the top 10K Wikipedia pages (Meta-Wiki Contributors, 2026), where only 15% of the 600K collected triples were deemed *Valid*; an in-depth analysis revealed that fixing property definitions was the easiest way to significantly increase the proportion of valid triples. Using our own local version of property definitions, we applied our approach and assessed its precision and recall on a set of 1.2K pre-filtered triples from 50 randomly selected entities for which two authors annotated the validity of all triples beforehand. The full description of the experiments is provided in (Andrade et al., 2026). Figure 2 shows the evaluation of filtering (V_{1-3}) and recovery (R_{1-3}). By applying the filtering, only 501 of the 1,243 original triples remain (V_3), but the precision of the selected triples goes up from 0.786 to 0.980. Adding the recovery step, the precision remains stable at 0.984, but 244 more triples are now valid, for a recall of 0.727 (R_3)

3 Implementation

The triple collection, validation, filtering and recovery steps were implemented as a series of Python functions, all available through a Colab notebook. In Block 1 of Figure 1, the user can query one or more entities, along with the maximum number of instances of the same property they want for each

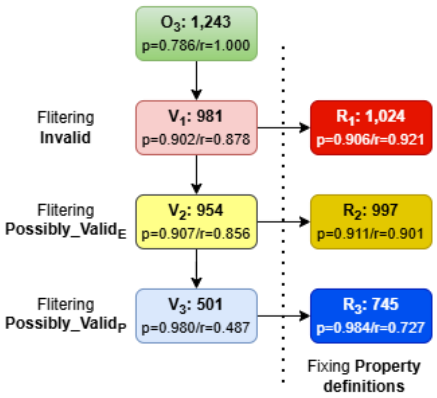


Figure 2: Triple count (top) and precision/recall (bottom) for each combination of filtering and recovery. O_3 is the pre-processed original data; V_{1-3} denote validation-based filters; R_{1-3} denote the recovery strategy applied to V_{1-3} respectively. From (Andrade et al., 2026).

entity. In Block 2, the user can choose one of three modalities: *Prioritise quantity* (highest recall in Fig. 2, cell O_3), *Prioritise quality* (highest precision in Fig. 2, R_3), or *Maximise both* (highest F1 in Fig. 2, R_1). Outputs can be produced in 4 formats: .csv, .json, .xml or .pkl.

4 Final considerations

Since the triple sets are highly customisable (type and number of entities, type and number of properties for each entities, etc.), it is always possible to create a new dataset that has not been brought together before. By selecting entities added after a cut-off date, it is also possible to create datasets with completely new contents. Once paired with aligned texts that exactly verbalise the triple sets, the collected data can be used not only for NLG but also for Information Extraction tasks.

Acknowledgments

Our work was funded by the Irish Department of Tourism, Culture, Arts, Gaeltacht, Sport and Media via the eSTÓR project, and has also benefitted more generally from being carried out in the wider context of the ADAPT SFI Centre for Digital Media Technology which is funded by Science Foundation Ireland through the SFI Research Centres Programme, and co-funded under the European Regional Development Fund (ERDF) through Grant 13/RC/2106.

References

Mark Andrade, Simon Mille, Anya Belz, and Brian Davis. 2026. Validating DBpedia triple sets for Natural Language Generation. *Preprint*, arXiv:2609.07589.

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.

Meta-Wiki Contributors. 2026. List of articles every Wikipedia should have/Expanded. `https://meta.wikimedia.org/wiki/List_of_articles_every_Wikipedia_should_have/Expanded`. Accessed: 2026-07-02.

Kristýna Onderková, Ondrej Platek, Zdeněk Kasner, and Ondrej Dusek. 2025. FreshTab: Sourcing fresh data for table-to-text generation evaluation. In *Proceedings of the 18th International Natural Language Generation Conference*, pages 108–121, Hanoi, Vietnam. Association for Computational Linguistics.