A Pipeline for Extracting Abstract Dependency Templates for Data-to-Text Natural Language Generation

Simon Mille,[1] **Josep Ricci,**[2] **Alexander Shvets**[2] and **Anya Belz**[1]
[1]ADAPT Research Centre, Dublin City University
[2]Pompeu Fabra University Barcelona
{simon.mille, anya.belz}@adaptcentre.ie
josep.ricci01@estudiant.upf.edu, alexander.shvets@upf.edu

Abstract

We present work in progress that aims to address the coverage issue faced by rule-based text generators. We propose a pipeline for extracting abstract dependency template (predicate-argument structures) from Wikipedia text to be used as input for generating text from structured data with the FORGe system. The pipeline comprises three main components: (i) candidate sentence retrieval, (ii) clause extraction, ranking and selection, and (iii) conversion to predicate-argument form. We present an approach and preliminary evaluation for the ranking and selection module.

1 Introduction

Rule-based Natural Language Generation (NLG) systems have become increasingly unpopular since the NLP field switched first to statistical systems, then to neural: rule-based systems tend to have low coverage (limited robustness to new inputs), reduced suprasentential fluency, and on the whole need to be built manually, all of which in combination means they are no longer competitive in shared task competitions and other NLP research contexts. However, their output can generally be guaranteed to have high accuracy and grammaticality, which continues to make them the system of choice in many commercial contexts.[1] Moreover, they can be efficient in terms of data and energy requirements, and suitable for low-resource languages. That is, on their own or in combination with, e.g., language-model-based modules, rule-based NLG potentially has an important role to play in the current NLP landscape if shortcomings such as the coverage issue addressed here can be overcome.

[1]E.g. Arria NLG's NLG Engine.

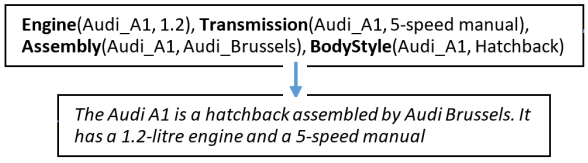

Figure 1: A DBpedia triple set from WebNLG+ and a corresponding generated text. Triple = Property(DB-Subj, DB-Obj), where the DB-Subj is an entity, and the DB-Obj another entity, a numeric, a date, etc.

WebNLG+. The present work was prompted by the WebNLG+ shared task (Castro Ferreira et al., 2020), in which part of the test set inputs contained features not seen in the training or development data. The WebNLG+ dataset is a benchmark for data-to-text NLG consisting of aligned DBpedia triple sets and texts. DBpedia triples are the building blocks of the inputs, and consist of three related elements called a *Property*, a *Subject* and an *Object* in Semantic Web terminology. A Subject (denoted by *DB-Subj* in this paper) is usually an entity that has a Property and a value for this Property, which is the Object (*DB-Obj*). E.g. in Figure 1, the entity *Audi_A1* is associated with 4 properties: *Engine*, *Transmission*, *Assembly* and *BodyStyle*. The semantics of each property is defined by DBpedia editors,[2] but in most cases, *the Property of the DB-Subj is DB-Obj* makes it clear (e.g., *the Transmission of the Audi_A1 is 5-speed manual*).

The coverage issue. Unlike their neural counterparts, rule-based generators submitted to the WebNLG+ challenge such as RDFJSREALB (Lapalme, 2020), DANGNT-SGU (Tran and Nguyen, 2020) or FORGe (Mille et al., 2019b) are not able to cope with new (previously *unseen*) properties. FORGe, which we are aiming to extend, operates on dependency structures at several levels of representation (syntax, semantics), and needs partially lexicalised predicate-argument (*PredArg*) structures in the PropBank style (Kingsbury and Palmer, 2002) to use as input for generation (see Figure 2b). In other words, if a mapping between

[2]See http://mappings.dbpedia.org/index.php/How_to_edit_the_DBpedia_Ontology.

Proceedings of the Seventh International Conference on Dependency Linguistics (Depling, GURT/SyntaxFest 2023), pages 91 - 101
March 9-12, 2023

property and PredArg structure as shown in Figure 2a-b does not exist, the generator cannot introduce the appropriate words and, unless a backup mechanism is in place, it will fail to generate a text.

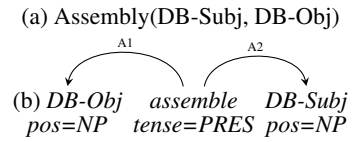

Figure 2: (a) The Property *Assembly*, (b) a corresponding PredArg template (graph with no linear order information), and (c) two possible verbalisations of the property *via* the template. A1/2 = first/second argument.

Thus, the overall problem that we are tackling is the following: given (i) the rule-based FORGe generator that covers all properties in the WebNLG+ training data, (ii) a file which contains the mappings between these properties and their respective PredArg template, (iii) an input triple set that contains one or more properties not currently covered by the generator, automatically extend the mapping file in ii with new unseen property/template pairs that will enable FORGe to generate a text that verbalises all input properties.

Proposed solution. Our aim is the automatic extraction of property/template pairs via a pipeline for retrieving and ranking candidate clauses from Wikipedia that correspond to a given DBpedia *instantiated property* (i.e. a triple), and converting them to predicate-argument representations. We are at an early stage of this research: the pipeline and components have been defined and connected, and we have identified two main challenges in our approach: one is candidate clause extraction, i.e. how to find a sentence or a clause that exactly matches the input triple, the other is the identification of such candidates, i.e. if provided with a list of candidates clauses that contains a match, is it possible to identify it. In this paper, we focus on the second challenge, since if we are not able to identify target candidates, the approach cannot work. In the remainder of the paper, we present the different components and resources used in our pipeline, and provide an encouraging preliminary quantitative and qualitative evaluation of a transformer-based candidate ranking and selection component.[3]

[3]The code and data are available at https://github.com/mille-s/PredArg-Template_Extraction.

2 Related Work

A number of papers have tackled the extraction of templates from text to be used as input for NLG. Duma and Klein (2013) mine and prune sentence templates from Wikipedia articles, but (i) extract templates given an entity (instead of a property as in our case), and (ii) manage to obtain a template for about 20% of the target entities. Ell and Harth (2014) achieve impressive coverage with their (multi-property) sentence templates, but also suffer accuracy problems, with the text faithfully representing the input in only about half the cases. Our general approach is conceptually similar to Perera and Nand (2015)'s, who use an open Information Extraction (IE) tool to identify candidate sentence spans that verbalise a given property, and then acquire lexicalisation information via VerbNet, resorting to default strategies when a predicate is not covered by VerbNet. Hoang et al. (2022) suggest several general approaches to align triple components and textual elements, namely string, substring, hypernym and synonym matching; for property matching, they also use a pre-trained vector model to calculate the distance between words. Other recent work on this topic uses keyword matching (Kaffee et al., 2022) or cosine similarity (Abhishek et al., 2022) for aligning triples and text in under-resourced languages. In order to assess the strength of the alignment, Abhishek et al. (2022) apply a Natural Language Inference (NLI) model to detect (lack of) entailment between the triples and the candidate sentences.

One difference between our approach and most of the related work on template mining for NLG is that we want to extract predicate-argument templates (Figure 2b), and not full-sentence templates. However, the approaches have a lot in common, since we extract the predicate-argument structures from sentences. The main issue with most of the approaches above is the lack of accuracy. Recently, Transformers have been shown to improve accuracy for Question-Answering (Karpukhin et al., 2020), including for the specific task of aligning text and structured data (Oguz et al., 2022) and also for fact checking, for instance for comparing tables and text (Zhang et al., 2020). In our approach, we therefore explore another way of aligning linguistic predicates and properties via Transformer-based meaning similarity scoring.

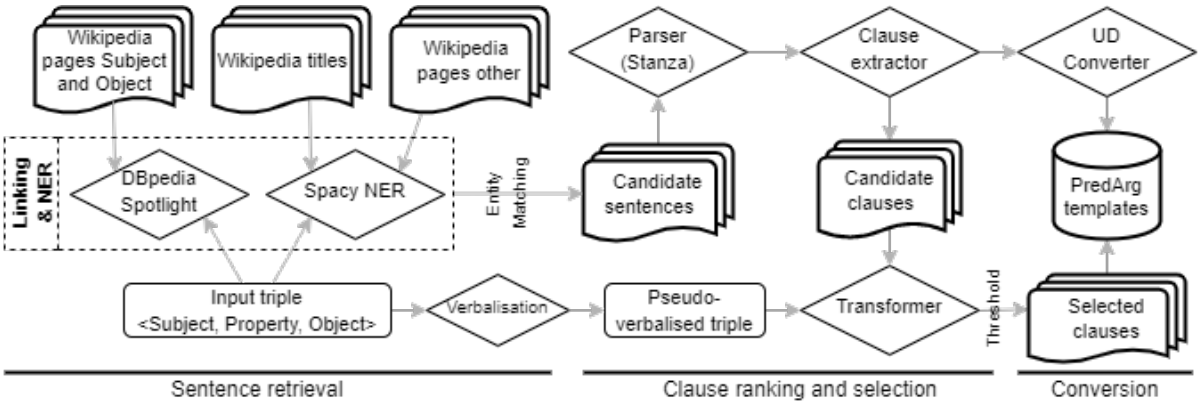

Figure 3: Overview of the pipeline for PredArg template extraction (see Appendix A for module output illustrations)

3 Template Extraction Pipeline

In this section, we describe the components that allows us to extract one or more PredArg template(s) given one input DBpedia triple.[4] Figure 3 shows a complete view of the pipeline (see Appendix A for module outputs). The three main components of the pipeline (indicated across the bottom in Figure 3) are: (i) Candidate sentence retrieval, (ii) Candidate clause ranking and selection, (iii) conversion to predicate-argument template.

3.1 Candidate sentence retrieval

The first step is to find candidate sentences for a given input triple; since DBpedia triples are often verbalised in Wikipedia texts, we use the Wikipedia contents as a candidate source. Via the HuggingFace dataset,[5] we have access to the title and the cleaned (plain) text of each article. We first find the Wikipedia articles of both the DB-Subj and the DB-Obj (if any), and then run the entity linking tool DBpedia Spotlight (Mendes et al., 2011) on the input triple's DB-Subj and DB-Obj and on the article text to find sentences that mention both the DB-Subj and the DB-Obj.

In order to find more candidate sentences and possibly get better candidates, we also perform a relaxed search. We get a named entity type for the DB-Subj using Spacy NER,[6] and parse Wikipedia article titles until we find an article about an entity of the same type as the DB-Subj. We then proceed to run Spacy NER on the DB-Obj and the found article so as to find sentences that contain two entities of the type of the DB-Subj and DB-Obj, and replace these entities with the ones from the original input for the ranking phase.

3.2 Candidate clause ranking and selection

In this section, we detail how we extract minimal clauses and calculate their semantic similarity with the input triple using a Sentence Transformer bi-encoder model[7] (Reimers and Gurevych, 2019), so that candidates scored above a given threshold are kept while others are discarded (see Section 4). Existing sentence similarity approaches return a score for a pair of sentences; in our case, we need a similarity score between a triple and a clause, so we fine-tuned the model to this task using a dataset created for this purpose.

Fine-tuning. We created a fine-tuning dataset with pseudo-verbalisations of input triples aligned with sentences from the WebNLG+ training set as follows. For each triple T, we compiled 4 sets of sentences that correspond to 4 levels of similarity with T: 1 (sentences that verbalise exactly T), 0.66 (sentences that verbalise a triple that has 2 elements in common with T, either DB-Subj and Property, DB-Subj and DB-Obj, or Property and DB-Obj), 0.33 (1 element in common with T), and 0 (no element in common with T), see Table 1.

We obtained 7,645 triple/sentence pairs in total for the set of similarity 1, 24K pairs for 0.66, 399K for 0.33 and 23M for 0. To balance the dataset, we randomly picked 7,645 pairs from the sets 0.66, 0.33 and 0. Finally, we converted each triple to a typed pseudo-verbalised form (Pasricha

[4] Since FORGe performs triple aggregation during the generation, we don't need to extract PredArg templates that correspond to multiple triples.

[5] https://huggingface.co/datasets/wikipedia

[6] https://spacy.io/api/entityrecognizer

[7] https://huggingface.co/sentence-transformers/nli-distilroberta-base-v2

Triple: Location(Agra Airport, India)	
1.00	'**Agra Airport is in India.**', '**Agra airport is located in India.**'
0.66	'**Agra Airport is located in** Uttar Pradesh.', 'The Taj Mahal **is in India.**', etc.
0.33	'AGR is the ATA Location Identifier for **Agra Airport.**', 'AC Hotel Bella Sky Copenhagen **is in** Copenhagen.', 'Mother Theresa is from **India**', etc.
0.00	'Agnes Kant is a national of the Netherlands.', 'FC Köln played the 2014-15 season in the Bundesliga.', 'Ampara Hospital has 476 beds.', etc.

Table 1: Sentences with different similarity levels; in **bold**, the elements in common with the triple.

et al., 2020): *Location(Agra Airport, India)* → *<AIRPORT> Agra Airport <PROP> location <PLACE> India*.[8] In our use case, when an unknown property is detected in the input, we will not have at hand a verbalisation of the triple that contains it since the objective of our pipeline is to discover such verbalisations. Therefore, the pseudo-verbalisation here is an adequate strategy: the pseudo-verbalised input triple will be compared to the candidate clauses.

Clause extraction. The sentences retrieved (see Section 3.1) are usually long, in the Wikipedia style; we thus reduce each sentence to the minimal subtree that contains a finite verb and two elements of the same types as the the DB-Subj and the DB-Obj respectively. Each candidate sentence is parsed with the Stanza Universal Dependency parser (Qi et al., 2020); the output syntactic structures are then processed to extract the minimal subtree via our own graph-transduction grammars. The original sentence span that corresponds to this clause subtree is selected (see Appendix A for illustration).

3.3 Conversion to PredArg templates

The predicate-argument structures of the selected clauses from the previous step are created. For this, we use the grammar-based UD Converter released for the Surface Realisation Shared Tasks (Mille et al., 2019a), which given a UD parse returns a predicate-argument structure. The specific DB-Subj and DB-Obj are replaced by generic [DB-Subj] and [DB-Obj] placeholders.

4 Experiments and preliminary results

In this paper, we provide a first evaluation of the ranking component; we believe that there are many ways of finding more candidate sentences (see Section 5), but predicting which candidate is suitable (or not) is particularly crucial in our pipeline.

[8] See Appendix B for details on the data and fine-tuning.

Evaluation setup. For the evaluation, we compare two models, the off-the-shelf Transformer (Reimers and Gurevych, 2019) and our fine-tuned version of it, on two datasets, (a) the WebNLG+ development subset of single-property inputs (401 triples), and (b) the subset of the WebNLG+ test set comprising all and only items with properties not seen in the WebNLG+ training data (113 triples). The objective is to obtain performance upper and lower bounds for the fine-tuned model by examining how accurate it is at selecting the right candidate (a) for properties seen during fine-tuning, and (b) for unseen properties, which is the most realistic scenario for PredArg template extraction. For each input triple, there are 1 to 3 exactly matching sentences (the corresponding reference sentences in the WebNLG+ dataset), which are the *target sentences* that we want the model to prefer (rank highest) for the input triple. For use as the non-matching candidates, which should be dispreferred (ranked lower) by the model, we select all other sentences that verbalise one-triple inputs, and all sentences that verbalise two-triple inputs; the total Dev and Test candidate pools contain 1,834 and 2,887 sentences respectively. This way, we ensure that we have candidates with a significant meaning overlap with the target sentences (one-triple inputs can share elements with one another, see Section 3.2, and two-triple inputs can include elements or even full triples of the one-triple inputs).

Results. On the development data (top half of Table 2), the fine-tuned model ranks all the target sentences at the top in 98.5% of the cases, and one of the target sentences at the top in 99.5% of the cases. The average similarity score of the correctly top-ranked sentences is 0.963, and the first non-target sentence is on average scored 0.346 points below. The off-the-shelf model is effective at placing one, but not all, target sentences at the top, and the difference in scores between the target and non-target sentences is half of what it is for the fine-tuned model (0.170 and 0.346 respectively).

To assess to what extend the models capture the semantics of the properties, we repeated the experiment above but modifying the input triples in two ways: replacing the property name by another randomly selected property (Avg. top P_{Mod}), and inverting the DB-Subj and DB-Obj (Avg. top P_{InvSO}). The off-the-shelf model has a harder time discriminating between correct and wrong properties than the fine-tuned model (similarity scores of

0.785 and 0.684, respectively, for the off-the-shelf model, 0.963 and 0.754 for the fine-tuned model). However, neither of the models is able to discriminate cases where the DB-Subj and DB-Obj are switched, yielding even higher scores on average than with the original triple (Avg. top P_{InvSO}).

We then looked for the threshold at which a model gets the best F1 score when selecting a candidate sentence. We tested all thresholds (in steps of 0.01 from 0 to 1) for each model on the Dev set and obtained values of 0.73 and 0.87 for the off-the-shelf and fine-tuned models respectively, which yield a F1 of 0.798 and 0.955 respectively. On the unseen test set, these thresholds yield a significantly lower F1 score, the fine-tuned model reaching an F1 of only 0.694 and the off-the-shelf model 0.429. Note that a better F1 can be achieved on these unseen triples by selecting different thresholds (both higher, at 0.93 and 0.78 respectively).[9]

Error analysis. We examined all the false positives and false negatives for the best threshold on the Dev set (0.87), and found the following errors.[10] *False positives (53 errors)*: (i) a sentence that corresponds to 2 triples was selected, because one or more elements of the second triple are very similar with the input triple's DB-Subj, DB-Obj or Property (75% of errors); (ii) the selected sentence verbalises a triple that is almost identical to the input triple (25%). *False negatives (35 errors)*: (i) mismatch between a DB-Subj, Property or DB-Obj and their corresponding verbalisation due to an accent, a comma in a number, quotation marks, parentheses, casing (57%); (ii) a triple element is verbalised with a word judged semantically distant (29%); (iii) a reference sentence is wrong (14%). Only false negatives (i) and (iii) in the stem from errors or lack of normalisation in the data; the other errors are due to the model.

Discussion. We were surprised by the decrease in the score between the Dev and the Test sets, especially for the off-the-shelf Transformer, for which we would expect no difference between seen and unseen properties. We hypothesise that the Test set is more challenging: (i) the reference sentences seem less similar (0.910 on Test VS 0.932 on Dev when running the off-the-shelf Transformer on the gold sentences for triples of size 1); (ii) some problematic cases are more frequent (e.g. the DB-Subj or DB-Obj has content in parentheses in 34% of the Test triples, VS 12% in the Dev set); (iii) there are more candidate sentences for the Test set (see Evaluation setup). There are likely other factors.

[9]Fig. 6 and 7 in Appendix C show the F1/Threshold plots.
[10]See Tables 3 to 8 in Appendix D for examples.

All properties of Dev. Set (401 triples)		
	Off-the-shelf	*Fine-tuned*
$Accuracy_{All}$ (%)	91.02	98.50
$Accuracy_{One}$ (%)	98.25	99.50
Avg. top P_{OK}	0.785	0.963
Margin	0.170	0.346
Avg. top P_{Mod}	0.684	0.754
Avg. top P_{InvSO}	0.803	0.971
F1 (thresh.)	0.798 (0.73)	0.955 (0.87)
Unseen porperties of Test Set (113 triples)		
	Off-the-shelf	*Fine-tuned*
$Accuracy_{All}$ (%)	56.64	73.45
$Accuracy_{One}$ (%)	87.61	96.46
Avg. top P_{OK}	0.787	0.929
Margin	0.110	0.212
Avg. top P_{Mod}	0.702	0.776
Avg. top P_{InvSO}	0.815	0.952
F1 Dev thresh.	0.429	0.694
F1 (best thresh.)	0.537 (0.78)	0.745 (0.93)

Table 2: Evaluation of the ranking module (WebNLG+). **Accuracy**$_{All/One}$ = % of cases with all/one good candidate(s) ranked at the top; **Avg. top** $\mathbf{P}_{OK}$ = Average score (0 to 1) of correctly top-ranked n candidates for a given input triple; **Margin** = difference in % between top ranked candidates and first non-correct candidate; **Avg. top** $\mathbf{P}_{Mod/InvSO}$ = Average score (0 to 1) of the top-ranked candidate for a given input triple in which the property name was randomly changed / the DB-Subj and DB-Obj were inverted; **F1**: best F1 score for candidate selection obtained via the indicated threshold.

5 Future work

We are currently developing the approach reported here further, including investigating how to increase the F1 for candidate selection on unseen data, for instance by using cross-encoders for the final ranking of the top candidates or NLI to filter out bad candidates (Abhishek et al., 2022). To find more and better candidates, we will apply co-reference resolution on the Wikipedia pages, test Open IE approaches to identify text spans (Perera and Nand, 2015), and explore the use of Simple Wikipedia (Duma and Klein, 2013) and WEXEA (Strobl et al., 2020). We will further develop our prototype clause extractor, and will apply our approach to other languages to test its portability.

Acknowledgements

This research was funded via (i) ADAPT/DCU by the MSCA-PF-EF 2021 grant awarded for the action 101062572, and (ii) UPF by the EC-funded research and innovation programme Horizon Europe under the grant agreement number 101070278 and by the Erasmus+ programme.

References

Tushar Abhishek, Shivprasad Sagare, Bhavyajeet Singh, Anubhav Sharma, Manish Gupta, and Vasudeva Varma. 2022. Xalign: Cross-lingual fact-to-text alignment and generation for low-resource languages. *arXiv preprint arXiv:2202.00291*.

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.

Daniel Duma and Ewan Klein. 2013. Generating natural language from linked data: Unsupervised template extraction. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)–Long Papers*, pages 83–94.

Basil Ell and Andreas Harth. 2014. A language-independent method for the extraction of RDF verbalization templates. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 26–34, Philadelphia, Pennsylvania, U.S.A. Association for Computational Linguistics.

Thang Ta Hoang, Alexander Gelbukh, and Grigori Sidorov. 2022. Mapping process for the task: Wikidata statements to text as wikipedia sentences. *arXiv e-prints*, pages arXiv–2210.

Lucie-Aimée Kaffee, Pavlos Vougiouklis, and Elena Simperl. 2022. Using natural language generation to bootstrap missing wikipedia articles: A human-centric perspective. *Semantic Web*, 13(2):163–194.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Paul Kingsbury and Martha Palmer. 2002. From TreeBank to PropBank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain. European Language Resources Association (ELRA).

Guy Lapalme. 2020. RDFjsRealB: a symbolic approach for generating text from RDF triples. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 144–153, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8.

Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, and Leo Wanner. 2019a. The second multilingual surface realisation shared task (SR'19): Overview and evaluation results. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)*, pages 1–17, Hong Kong, China. Association for Computational Linguistics.

Simon Mille, Stamatia Dasiopoulou, and Leo Wanner. 2019b. A portable grammar-based nlg system for verbalization of structured data. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1054–1056. ACM.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2022. UniK-QA: Unified representations of structured and unstructured knowledge for open-domain question answering. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1535–1546, Seattle, United States. Association for Computational Linguistics.

Nivranshu Pasricha, Mihael Arcan, and Paul Buitelaar. 2020. NUIG-DSI at the WebNLG+ challenge: Leveraging transfer learning for RDF-to-text generation. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 137–143, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Rivindu Perera and Parma Nand. 2015. A multi-strategy approach for lexicalizing linked open data. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 348–363. Springer.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. 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.

Michael Strobl, Amine Trabelsi, and Osmar Zaiane. 2020. WEXEA: Wikipedia EXhaustive entity annotation. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1951–1958, Marseille, France. European Language Resources Association.

Trung Tran and Dang Tuan Nguyen. 2020. WebNLG 2020 challenge: Semantic template mining for generating references from RDF. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 177–185, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Hongzhi Zhang, Yingyao Wang, Sirui Wang, Xuezhi Cao, Fuzheng Zhang, and Zhongyuan Wang. 2020. Table fact verification with structure-aware transformer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1624–1629, Online. Association for Computational Linguistics.

A Sample outputs of all components

In this section , we illustrate each step of the whole pipeline.

Input triple
Alan_Bean || birthDate || "1932-03-15"

Entity linking (DBpedia Spotlight)

- DB-Subj: Alan Bean
  - kb$_{id}$: 11139903761698166211
  - dbpedia link: http://dbpedia.org/resource/Alan_Bean
- DB-Obj: "1932-03-15"
  - kb$_{id}$: 0 (No dbpedia entity found)

Entity type assignment (Spacy)

- DB-Subj: Alan Bean
  - Entity label: 380 (PERSON)
- DB-Obj: "1932-03-15"
  - Entity label: 391 (DATE)

Typed pseudo-verbalisation

We first check if the DB-Subj or DB-Obj are a number –using regular expressions- or a time period –using the python module dateutil.parser. If not, we do the DBpedia query:

- DB-Subj (Alan_Bean): None
- DB-Obj (1932-03-15): TIMEPERIOD

Since the DB-Obj has a type, we then query DBpedia for the DB-Subj only, and choose the first returned type (in bold below):

{{'uri': '**http://dbpedia.org/ontology/Person**'},
{'uri': 'http://dbpedia.org/ontology/Animal'},
{'uri': 'http://dbpedia.org/ontology/Astronaut'},
{'uri': 'http://dbpedia.org/ontology/Eukaryote'},
{'uri': 'http://dbpedia.org/ontology/Species'}}

We can then proceed to produce the pseudo-verbalised triple as follows:

<PERSON> Alan Bean <PROP> birth date <TIMEPERIOD> "1932-03-15"

Sentence extraction (Entity matching) and parsing (Stanza)
To get Wikipedia pages, we retrieve (i) the page of the DB-Subj, (ii) the page of the DB-Obj if any, and (iii) 1,000 random article about an entity that has the same type as the DB-Subj (matching the Spacy tag of the title with that of the DB-Subj). We then look for candidates on the pages, based on the type predicted by DBpedia Spotlight (pages of DB-Subj and DB-Obj) or by Spacy (other pages). We detokenise the DB-Subj and the DB-Obj for them to be parsed as one single named entity.

1	The	DT	Definite=Def|PronType=Art	2	det
2	seat	NN	Number=Sing	11	nsubj
3	of	IN	_	5	case
4	Wheeler	NNP	Number=Sing	5	compound
5	County	NNP	Number=Sing	2	nmod
6	,	,	_	2	punct
7	in	IN	_	8	case
8	Texas	NNP	Number=Sing	5	nmod
9	,	,	_	11	punct
10	is	VBZ	Mood=Ind|...	11	cop
11	Wheeler	NNP	Number=Sing	0	root
12	,	,	_	11	punct
13	where	WRB	PronType=Rel	16	mark
14	Alan_Bean	NNP	subject=true	16	nsubj:pass
15	was	VBD	Mood=Ind|...	16	aux:pass
16	born	VBN	Tense=Past|...	11	acl:relcl
17	on	IN	_	18	case
18	1932-03-15	CD	NumForm=Digit|...	16	obl
19	.	.	_	11	punct

Figure 4: Sample UD structure (selected columns)

Clause Extraction (graph transduction grammars)
The output of the clause extractor is the minimal subtree that contains both the DB-Subj and the DB-Obj, with additional trimming (e.g. a relative pronoun before the DB-Subj is removed): 'Alan_Bean was born on "1932-03-15"'

Clause ranking (Transformer)
The similarity of the extracted clause with the input triple is then calculated: 'Alan_Bean was

born on "1932-03-15"' -> 0.8853045701980591'. If the clause is above the defined threshold, it is selected for the template. See more examples of ranking and selection in Appendix D.

Conversion to PredArg (UD Converter)
Figure 5 shows the delexicalised predicate-argument template extracted from the selected clause.

```
1       bear    VERB    Tense=Past|...  0       ROOT
2       [Subject]       PROPN   subject=true|...        1       A2
3       [Object]        NUM     NumForm=Digit|...       1       Time
```

Figure 5: Sample PredArg template (selected columns)

B Details on the fine-tuning step

Our method for triple pseudo-verbalization is based on the one in (Pasricha et al., 2020); we adapted a couple of aspects not detailed in the paper: (i) we implemented our own simple functions for checking if a DB-Obj is of type number or date, and (ii) we took the first ontology type (starting with *dbo:*) in the *rdf:type* section of the DBpedia page for the other types.

The finetuning dataset is built from the one-triple items in the test set of the WebNLG+ dataset.[11] For finetuning the model, we sample 7,645 items for each of the 4 similarity categories as explained in the paper. The sample is divided 70/15/15 for training, development and test sets, respectively. The train batch size is 16, and the train loss is Cosine Similarity Loss. It uses the Embedding Similarity Evaluator (which uses the development set) with evaluation steps = 1000, and some warm-up steps (10% of the training data), with num_epochs = 4.

C Plots F1-score clause ranking and selection

Figures 6 and 7 show a plot of the F1-score in function of the selection threshold for candidate sentences.

D Sample classification errors

Tables 3 to 8 show examples of mis-selection of candidate sentences for an input triple. In cyan, correctly selected target sentences; in orange, erroneously selected (false positive) or discarded (false negative) sentences.

[11] https://drive.google.com/file/d/1BM-W0GTa931jdNp1De_vHcfa8GGdPhTL/view?usp=sharing

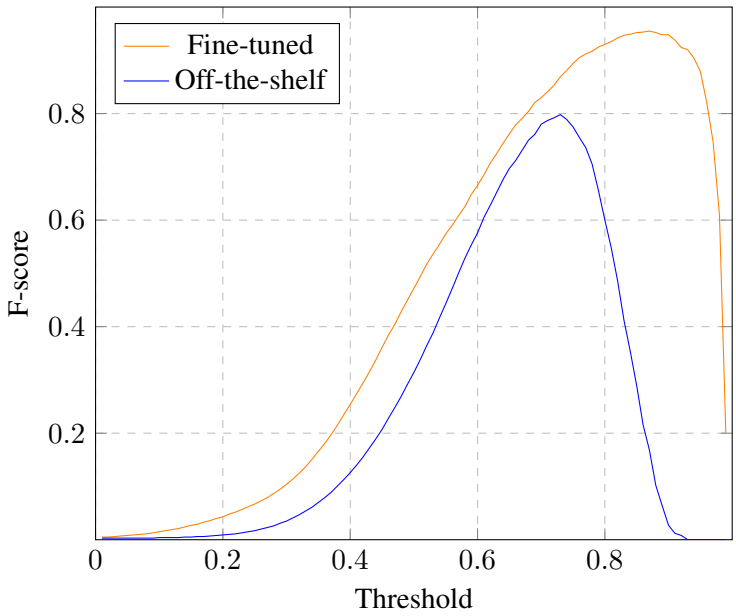

Figure 6: Threshold definition for clause selection (Development set)

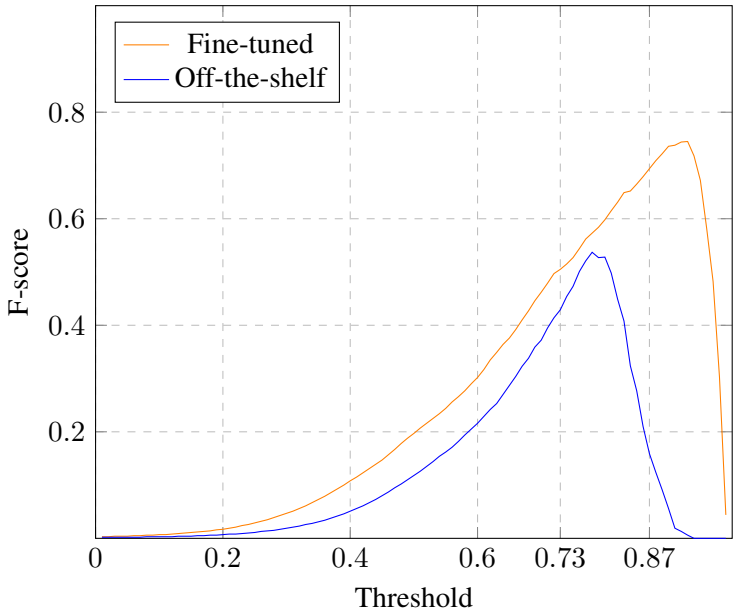

Figure 7: F1 score for clause selection (Test set)

Input
<AIRPORT> Athens International Airport <PROP> location <PLACE> Spata
Target sentences
Athens International Airport is located in Spata.
Athens International Airport is in Spata.

Top-ranked sentences	Score
Athens International Airport is located in Spata.	0.958
Athens International Airport is in Spata.	0.955
Athens International Airport, which is located in Spata, serves the city of Athens.	0.949
Athens International Airport is in Spata and serves the city of Athens.	0.943
Athens International Airport in Spata serves the city of Athens.	0.934
Agra Airport is in Agra.	0.523

Table 3: False positive Dev Type (i) (Non-target sentence > 0.87)

Input
<PLACE> Ann Arbor, Michigan <PROP> leader title <PERSONFUNCTION> Mayor
Target sentences
Mayor, is the title of the leader in Ann Arbor, Michigan.
The leader title of Ann Arbor, Michigan, is Mayor.
Ann Arbor, Michigan is led by the Mayor.

Top-ranked sentences	Score
The leader title of Ann Arbor, Michigan, is Mayor.	0.994
Ann Arbor, Michigan is led by the Mayor.	0.990
Mayor, is the title of the leader in Ann Arbor, Michigan.	0.988
The City Administrator leads Ann Arbor in Michigan.	0.908
A City Administrator leads Ann Arbor, Michigan.	0.897
Albany, Georgia is led by a Mayor.	0.657

Table 4: False positive Dev Type (ii) (Non-target sentence > 0.87)

Input
<AIRPORT> Alpena County Regional Airport <PROP> runway length <NUMERIC> 1533.0
Target sentences
The runway length of Alpena County Regional Airport is 1,533.
The runway length of Alpena County Regional airport is 1533.0.

Top-ranked sentences	Score
The runway length of Alpena County Regional airport is 1533.0.	0.995
The runway length of Alpena County Regional Airport is 1,533.	0.567
The Adolfo Suárez Madrid–Barajas Airport is in San Sebastián de los Reyes and has a runway length of 3500.0 metres.	0.474
Located in Alcobendas, Adolfo Suarez Madrid-Barajas Airport has a runway with the length of 3500.0 metres.	0.470
The Adolfo Suárez Madrid–Barajas Airport located at San Sebastian de los Reyes has a runway length of 3500.	0.466
Ann Arbor, Michigan has a population of 1580.7 per square kilometre and a total area of 74.33 square kilometres.	0.464

Table 5: False negative Dev Type (i) Number (Target sentence < 0.87)

Input
<FOOD> Bakso <PROP> ingredient <FOOD> Noodle
Target sentences
Bakso contains noodles.
Noodle is an ingredient in Bakso.
The dish Bakso contains noodles.

Top-ranked sentences	Score
Noodle is an ingredient in Bakso.	0.989
The dish Bakso contains noodles.	0.857
Bakso contains noodles.	0.820
Vermicelli is an ingredient in Bakso.	0.640
Vermicelli is an ingredient of the dish Bakso.	0.636
Vermicelli is included in bakso.	0.553

Table 6: False negative Dev Type (i) Casing (Target sentence < 0.87)

Input
<PERSON> N. R. Pogson <PROP> nationality <MUSICALARTIST> England
Target sentences
N. R. Pogson was English.
N.R. Pogson was an English national.
N. R. Pogson is British.

Top-ranked sentences	Score
N.R. Pogson was an English national.	0.913
N. R. Pogson is British.	0.909
N. R. Pogson was English.	0.574
People from the United Kingdom are called British people.	0.482
British people is a demonym for people in the United Kingdom.	0.458
The native people of the United Kingdom are known as the British people.	0.441

Table 7: False negative Dev Type (ii) (Target sentence < 0.87)

Input
<PLACE> Swords, Dublin <PROP> is part of <SETTLEMENT> Dublin (European Parliament constituency)
Target sentences
Swords is a part of the Dublin European Parliamentary constituency.
Swords belongs to the Dublin constituency of the European Parliament.
Swords, Dublin is part of the Dublin European Parliament constituency.

Top-ranked sentences	Score
Swords, Dublin is part of the Dublin European Parliament constituency.	0.893
Swords is a part of the Dublin European Parliamentary constituency.	0.835
Swords belongs to the Dublin constituency of the European Parliament.	0.774
Trane is located in Swords, Dublin, Ireland.	0.638
Trane is located in Swords, Dublin, which is in Ireland.	0.625
The location of Trane is in Swords, Dublin, Ireland.	0.620

Table 8: False negative Dev Type (iii) (Target sentence < 0.87)