Fine-Tuning, Prompting and RAG for Knowledge Graph-to-Russian Text Generation. How do these Methods generalise to Out-of-Distribution Data?

Anna Nikiforovskaya, William Soto-Martinez, Evan Chapple and Claire Gardent
CNRS/LORIA and Université de Lorraine (France)
firstname.lastname@loria.fr

Abstract

Prior work on Knowledge Graph-to-Text generation has mostly evaluated models on in-domain test sets and/or with English as the target language. In contrast, we focus on Russian and we assess how various generation methods perform on out-of-domain, unseen data. Previous studies have shown that enriching the input with target-language verbalisations of entities and properties substantially improves the performance of fine-tuned models for Russian. We compare multiple variants of two contemporary paradigms — LLM prompting and Retrieval-Augmented Generation (RAG) — and investigate alternative ways to integrate such external knowledge into the generation process. Using automatic metrics and human evaluation, we find that on unseen data the fine-tuned model consistently underperforms, revealing limited generalisation capacity; that while it outperforms RAG by a small margin on most datasets, prompting generates less fluent text; and conversely, that RAG generates text that is less faithful to the input. Overall, both LLM prompting and RAG outperform Fine-Tuning across all unseen testsets. The code for this paper is available at https://github.com/Javanochka/KG-to-text-fine-tuning-prompting-rag

1 Introduction

Generating from structured data has a variety of applications such as verbalising tabular data (Lebret et al., 2016; Sha et al., 2018), question answering from databases (Fan et al., 2019) and generating from meaning representations (Flanigan et al., 2016; Marcheggiani and Perez-Beltrachini, 2018; Ribeiro et al., 2019; Zhu et al., 2019; Fan and Gardent, 2020) or from knowledge graphs (Marcheggiani and Perez-Beltrachini, 2018).

We consider Knowledge Graph-to-Text (KG-to-Text) generation where the task consists of verbalising the content of a KG, and a KG is a set of triples of the form (subject, property, object) where subject are entities and objects are values or entities. While KG-to-Text generation has been extensively explored in the context of the WebNLG shared tasks (Gardent et al., 2017; Castro Ferreira et al., 2020a; Cripwell et al., 2023) , in this work, we investigate the generalisation capacity of three main generation methods (fine-tuning, LLM prompting and Retrieval Augmented Generation) taking Russian as a target language and evaluating on Out-Of-Domain (OOD) datasets.

Compared to generation into English, generating from knowledge graphs into Russian faces the challenge of decoding into a language with a different script, a rich verb morphology and a complex nominal declension system. Prior work shows that lexicalising entities into Russian is a difficult task (Chuklin et al., 2022) and that pre-trained models fine-tuned on the Russian WebNLG dataset show a strong performance decrease on non i.i.d data (Nikiforovskaya and Gardent, 2024).

Starting with the state-of-the-art KG-to-Russian models presented in (Kazakov et al., 2023), we begin by showing that a pre-trained model fine-tuned on the WebNLG data under-performs on both seen and unseen data but that the same model achieves much better results when the input graph is enriched with Russian lexicalisations of the entities and properties present in the input graph.

Next we explore various LLM-based, Few-Shot Prompting approaches (Brown et al., 2020). Generating with an LLM benefits from the parametric knowledge acquired from their training on large amounts of data. Compared to (Kazakov et al., 2023)'s model which is a fine-tuned version of T5 (Raffel et al., 2023), and therefore trained on much smaller quantities of data than LLMs, the knowledge encapsulated in an LLM might help generalise to unseen test data. As in the fine-tuning approach described above, we enrich the input prompt with Russian lexicalisations (Wikidata labels) of the entities and properties present in the input graph, and we experiment with various ways of selecting few-

Proceedings of the 18th International Natural Language Generation Conference, pages 419–448
October 29–November 2, 2025. ©2025 Association for Computational Linguistics

shots. Intuitively, the selected few shots should help provide two types of information: (i) how the properties and entities present in the input graph are verbalised into Russian and (ii) how a knowledge graph is verbalised into a text. Accessing these two sources of knowledge can be viewed as learning both entity/property verbalisation and a generation template. Sample templates can be identified from the associated text and property verbalisation by jointly accessing graph properties and their textual mention. We introduce a few-shot selection method which systematically improves generation results by selecting few shots whose graph size and properties resemble the size and properties present in the input graph.

Retrieval Augmented Generation (Gao et al., 2024), a method which extends the generation input with retrieved information, is another common way to integrate external knowledge into the generation process. Different from the other two generation approaches we explored, we use retrieval, not to extract Wikidata labels, but to retrieve textual data from Wikipedia. This removes the need for a knowledge base that provides multilingual labels for entities and properties and enables an approach that can more easily adapt to other knowledge bases. We explore a variety of strategies for querying, chunking and few-shots selection and we show that the best performing RAG variant performs almost on par with the LLM-prompting approach while being less restrictive in terms of resources.

In summary, we compare three ways of enriching the generation input to help the model generalise to unseen input:

- Fine-Tuning: we enrich the input of a small (1.7B parameters) pre-trained model with Russian lexicalisation of entities/properties which are extracted from Wikidata. We hypothesize that while the additional information should improve generation results on seen data, the limited capacity of the small pre-trained model restricts its ability to generalize to out-of-domain test sets.

- LLM Prompting: we include Russian names for KG entities and properties in the prompt of an LLM. We seek to examine (i) how the incorporation of additional information affects the generative performance of large language models and (ii) whether the parametric knowledge encapsulated by LLMs better supports generalization to out-of-domain (OOD) data.

- Retrieval Augmented Generation: we aim to guide text generation using passages retrieved from the Russian Wikipedia that are semantically aligned with the information conveyed by the input graph. This approach eliminates dependence on knowledge base labels, providing a flexible framework that can readily adapt to other knowledge bases. We investigate its performance relative to LLM prompting and fine-tuning on both seen and unseen datasets.

2 Related Work

Various approaches have been proposed to convert KGs into text including grammar- and template-based approaches, custom encoder-decoders, fine-tuned pre-trained models and pipelines (Gardent et al., 2017; Castro Ferreira et al., 2020b; Cripwell et al., 2023). More recently, LLM prompting has also been explored. Yuan and Faerber (2023) evaluate ChatGPT on the AGENDA and WebNLG test sets - they find that both models perform relatively poorly on most measures. Axelsson and Skantze (2023) evaluate GPT-3.5-turbo on both the WebNLG and on some non i.i.d. test sets they created, showing good results on English in-domain data but low performance on Russian and poor semantic adequacy on OOD data. Comparing zero- and few-shot variants of GPT3.5-Turbo, LLaMA-7B, Vicuna-7B and LlaMA-7B fine tuned on WebNLG data, Schneider et al. (2024) finds that fine-tuning yields the best results on WebNLG test data. Finally, He et al. (2025) show that selecting few-shots based on complexity and diversity enhance results for KG-to-English generation. Other work has also explored Retrieval Augmented Generation. In particular, Jobanputra and Demberg (2024) propose a few-shot RAG system for English where few shots are selected to match the properties contained in the input graph.

Leveraging the WebNLG training and test data, some work has explored generation into Russian. Agarwal et al. (2020) pre-train and fine-tune T5 on several parallel corpora in a multi-task (NLG/Semantic parsing) setting, obtaining best results in the 2020 Shared Task. Mille et al. (2024) use a rule-based approach to convert triples into English, a language model to paraphrase the resulting texts and a Machine Translation model to translate English into Russian. On the WebNLG testsets, the best results for generation into Russian are obtained by fine-tuned models, with Kumar et al. (2023) presenting a fine-tuned mT5 model (Xue et al., 2021)

and Kazakov et al. (2023) achieving the state of the art by fine-tuning the FRED-T5 model (Zmitrovich et al., 2024) on the WebNLG training data for Russian.

Other work has explored the behaviour of KG-to-Text models on OOD data. In particular, Xu et al. (2023) assess generalisation by constraining the training data and evaluating on the full WebNLG testsets. They show that the SoTA pre-trained language models (Raffel et al., 2020; Kale and Rastogi, 2020) fail to generalize and that a compositional approach based on clustering helps improve generation. Mille et al. (2021) derive challenge testsets from the existing WebNLG data to identify performance decrease that would associate with specific linguistic phenomena (e.g., large number of co-referring entities) or graph characteristics (e.g., high number of triples). Kasner and Dusek (2024) show that, on unseen data which they created by collecting data records from public APIs, more than 80% of the output contain at least one semantic error.

We depart from previous work in that we focus on Russian, evaluate on unseen testsets and compare the behaviour of fine-tuned models, LLM-prompting and RAG on this unseen data.

3 Data and Metrics

3.1 Test Data

We evaluate our models on three datasets of (KG, Russian Text) pairs.

Seen (in-domain) data. This data is available on the web and is might therefore have been included in the LLM training data. The dataset consists of the 1,102 instances used for the evaluation of the WebNLG 2020 and 2023 campaigns. In this dataset, the KGs contain seen entities and predicates i.e., entities and predicates that are present in the Russian WebNLG training data.

Unseen (Out-of-Domain) Data. This data set is not available on the web. It was created by (Nikiforovskaya and Gardent, 2024) and derived from the WebNLG and KELM (Agarwal et al., 2021) datasets.

From the WebNLG dataset, two datasets are derived. The Unseen Category (WebNLG-C, 1,251 instances) contains WebNLG graphs from the English training data whose root DBPedia category does not belong to the set of categories present in the Russian WebNLG training data. In contrast, the Unseen Entity testset (WebNLG-E, 192 instances) contains graph instances that are from seen categories (i.e., graphs whose root category belong to one of the 16 categories present in the Russian WebNLG train or dev set) but whose entities are unseen.

KELM is a silver dataset of (Wikidata graph, English text) pairs created using distant supervision. Similar to the WebNLG unseen testsets, Nikiforovskaya and Gardent (2024) derives two gold test sets from KELM: a dataset where all graph properties are present in the WebNLG training/dev data for Russian but some entities are not (KELM-E, 2126 instances) and a dataset whose graphs contain both unseen entities and unseen properties (KELM-E+P, 1311 instances).

Unseen (Out-of-Domain) New Graphs Data. For both KELM and WebNLG, the graphs are available on the web and might have been included in the LLM training data. We create a dataset of 472 (Unseen New Graphs, Russian Text) pairs by extracting subgraphs from Wikidata and manually editing the Russian texts that were automatically generated from these graphs by our best LLM prompting approach (Cf. Section 5, prompting configuration PRU/+L/D.). The graphs were constructed by first identifying non-existing or rare properties in KELM and WebNLG; manually deciding on relevant Wikidata categories (i.e., categories where subsets of these properties often co-occur); defining a set of possible graph schemas for each category; and using these schemas to extract matching subgraphs from Wikidata. This new dataset consists of 472 (Wikidata graphs, Russian text) pairs with graphs containing from 1 to 10 triples. The categories are Airlines, Chemicals, Conferences, Statistics, Dancers, Diseases, and Tournaments. Each of these categories contains graphs of 1 to 7 triples. We also combined some of the graphs from Statistics and Airlines categories to get a Mixed category containing only graphs of size 10. This testset is further called NG_{500}. As a result of its construction, out of 50 properties existing in NG_{500} only 3 exist also in WebNLG training data; ‘occupation‘, ‘inception‘ and ‘country‘. While ‘inception‘ is a rare property, ‘occupation‘ and ‘country‘ are used in new contexts compared to WebNLG, so we allowed their presence in NG_{500} graphs.

In table 1 we provide statistical information for each testset we further use.

Dataset	# graphs	# lexs	# ents	# props
WebNLG Seen	1102	2779	1158	192
WebNLG E	192	552	56	41
WebNLG C	1251	3632	531	151
KELM-E	2126	2126	4038	53
KELM-E+P	1311	1311	4073	296
NG_{500}	472	472	1611	50

Table 1: Statistical information on each of the used test-sets, number of graphs, lexicalisations, unique entities and unique properties.

3.2 Fine-Tuning and Few-Shot Data

For fine-tuning, we use the WebNLG training data for Russian, an aligned corpus of 6,339 (KG, Russian text) pairs. This dataset, together with the English WebNLG training data, also serves as the retrieval basis for few-shot selection.

3.3 Metrics

For automatic evaluation, we use the following metrics: BLEU score (Papineni et al., 2002), chrF++ (Popović, 2017), TER score (Snover et al., 2006) and BERT score (Zhang et al.).

We also carry out a human evaluation (cf. Section 7.2) to compare the best setting of each of the three methods.

4 Fine-tuning on Labeled data

The first method we explore builds on the state-of-the-art Interno KG-to-text model for Russian (Kazakov et al., 2023), which fine-tunes FRED-T5 model on the WebNLG training data [2].

To facilitate entity verbalisation, the authors modify the input in the training and test sets to include, in addition to the knowledge graph, the Russian entity and property labels which are provided by the WebNLG data. As our unseen test sets do not contain such labels, we extract Russian entity and property labels from the Wikidata Knowledge Base using the SparQL queries shown in Appendix B. Retrieved labels are added to the generation input under the key "additional links" using the format "English_name=Russian_name".

We then run the Interno model on both seen and unseen data, comparing results when the input is and is not enriched with Russian labels.

Model	BLEU		chrF++	B_{F1}
WebNLG Seen				
Interno	24.59	(B)	0.44	0.82
Interno+labels [1]	**53.67**	(A)	**0.69**	**0.92**
WebNLG E				
Interno	22.43	(A)	0.48	0.85
Interno+labels	23.58	(A)	0.50	0.86
WebNLG C				
Interno	23.02	(B)	0.42	0.83
Interno+labels	**24.62**	(A)	**0.46**	**0.84**
KELM-E				
Interno	17.57	(A)	0.36	0.80
Interno+labels	17.74	(A)	0.44	0.84
KELM-E+P				
Interno	12.39	(B)	0.36	0.78
Interno+labels	**16.10**	(A)	**0.45**	**0.81**
NG_{500}				
Interno	14.93	(A)	0.34	0.77
Interno+labels	14.75	(A)	0.36	0.79
All Unseen				
Interno	17.52	(B)	0.37	0.80
Interno+labels	**18.89**	(A)	**0.44**	**0.83**
All				
Interno	18.72	(B)	0.39	0.81
Interno+labels	**24.83**	(A)	**0.49**	**0.84**

Table 2: **Results of (Kazakov et al., 2023)'s Pretrained Model fine-tuned on WebNLG 2020 training data, tested with and without input enrichment.** We provide BLEU score, chrF++ and BERT F1 score. The letters A/B indicate whether the difference between two scores occurring in the same cell is or not statistically significant. We assess statistical significance using Paired Bootstrap Resampling as described in (Koehn, 2004) and consider a difference as significant if confidence is at least 90%.

Results. Table 2 shows the results and Table 7 in the Appendix indicates the ratio of entities and properties present in the input graphs for which a Wikidata label could be retrieved. The results clearly show that, when a sufficient ratio of labels can be retrieved, enriching the input with Russian labels improves performance. Thus, for the three datasets where the ratio of retrieved labels is the lowest (NG_{500}: 12%, K_E: 27%, W_E: 41%), the improvement at generation time is not statistically significant while for the other three datasets where this ratio is higher (W_C: 43%, W_S: 46%, K_{E+P}: 56%), it is.

5 LLM Prompting

While the previous section demonstrated that enriching the input with entity and property labels

[1] Re-running the authors code on this data set using the DBPedia labels as they did, produced scores similar but not identical to the scores presented in the author's paper.

[2] FRED-T5 (1.7B parameters) is based on the T5 architecture (Raffel et al., 2023) and trained on a large corpus of Russian texts using a mixture of denoising objectives (Zmitrovich et al., 2024).

	Seen	Unseen				
	WebNLG	WebNLG		Kelm		NG_{500}
FS/I/NLG	W_S	W_E	W_C	K_E	K_{E+P}	
Interno+labels	53.67	23.58	24.62	17.74	16.10	14.75
Baseline						
R/G/D	16.86	24.38	25.08	20.50	16.16	8.80
Input variants						
R/+P/D	16.79	26.58	24.88	20.52	14.31	8.17
R/+L/D	20.53	25.47	25.88	21.88	16.45	8.01
R/+All/ D	14.85	22.69	22.12	17.94	14.18	5.60
FS variants						
S/G/D	16.91	24.75	23.53	19.18	16.40	8.90
P/G/D	19.76	25.48	23.64	18.17	16.88	8.47
NLG variant						
R/G/ Cum	15.62	21.94	22.55	19.88	13.84	18.16
Combined						
P/+L/D	23.16	23.75	25.04	19.80	**18.33**	8.34
P/+P+L/D	22.77	24.11	24.76	20.52	16.32	6.75
Russian Few Shots						
PRU/+L/D	**32.15** (A)	**30.09** (A)	**25.76** (A)	21.36 (A)	16.19 (A)	20.54 (B)
PRU/G/D	30.81 (B)	29.76 (A)	24.76 (B)	21.54 (A)	16.51 (A)	**21.51** (A)

Table 3: **BLEU Scores for the various LLM-Prompting configurations (FS/I/NLG).** FS specifies the few shot selection strategy (R:Random, S:Size, P:Property overlap on fewshots selected from the English WebNLG data, PRU: Property overlap on fewshots selected from the Russian WebNLG data); I, the input (G:Graph, +P:Input augmented with properties descriptions, +All:Input augmented with properties and entities descriptions, +L:Input augmented with Russian labels) and NLG, the generation strategy (D:Direct, Cum:Cumulative). Underlined scores are best score within a block, boldface indicates best score for column.

improves performance, we find that the gains are most pronounced on seen data. We hypothesize that the parametric knowledge encoded in larger generative models could enhance generalization, and we therefore explore LLM prompting.

For LLM-Based generation, we use Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and experimented with a few-shots prompting strategy that varies across three dimensions: the input, the prompting strategy and how few-shots are selected. Details about each of these dimensions and example prompts, illustrating the various options for each dimension are given in Appendix A and subsection A.4.

Briefly, *the input* is either the knowledge graph (G) or the KG enriched with Wikidata property descriptions (+P), Wikidata Russian entity labels (+L), Wikidata Russian property descriptions and entity labels (+P+L) or Wikidata property and entity descriptions (+All). The *prompting strategy* is to either prompt the LLM for a direct verbalisation of the entire graph (D) or to first verbalise each fact in the input graph and then aggregate the cumulative verbalisations into a fluent text (Cum). For *few shot selection*, we select 4 few shots[3] and explore three strategies: random (R); size-based, where we select few shots containing graphs whose size is the same as the size of the input graph (S)[4]; and property based, which consists in selecting few shots whose graphs contain properties similar to the set of properties present in the input graph (P when the selected few-shots are from the English retrieval base; PRU, when they are in Russian). Algorithm 1 in the Appendix shows the algorithm used for P and PRU few-shot selection.

Results. In Table 3 (*Input Variants*), we see that including Wikidata labels in the prompt (R/**+L**/D) helps improve generation, confirming the results of the previous section. Table 3 (*NLG Variants*) shows that, except on NG_{500}, the cumulative approach generally underperforms direct generation. A look at the data reveals that, different from the other testsets, the NG_{500} graphs do not require complex linguistic aggregation and that therefore for

[3]We experimented with 0, 2 and 4 few shots and found that 4 gave best results.

[4]Since the WebNLG data from which we retrieve the few shots has a maximum graph size of 7, when the input graph is larger we select few shots of size 7

this dataset, a simple modifications of the clauses verbalising the input triples often suffice to generate a text that matches the reference. We provide an example of such a graph in Appendix E.

The *FS Variants* block indicates that none of the few shot selection methods systematically improve results. However, Table 3 (*Combined*) shows that combining property-based few shot selection with an input enriched with labels yields good results overall. Finally, the *Russian Few Shots* variants indicate that augmenting the input with Wikidata labels and selecting Russian rather than English few-shots yields best results for all test sets except for NG_{500}, which, as observed in the previous section is the test set for which the ratio of retrieved labels is the lowest which in turn decreases the impact of adding labels to the input.

In sum, we find that the best prompting configuration (PRU/+L/D) is a configuration where the input graph is enriched with Wikidata entity and property labels and the selected few-shots are example graph/text pairs whose texts are in Russian and whose graphs have maximum similarity in terms of size and properties with the input graph.

6 Retrieval Augmented Generation for LLM

Unlike the previous two approaches we explore, the RAG method does not rely on Russian labels from a knowledge base. Instead, it retrieves semantically aligned Wikipedia text to enrich the input KG, providing a flexible framework that can readily adapt to other knowledge bases.

RAG provides a natural way to enrich the input by retrieving additional data from a vector base containing task-relevant knowledge and adding the retrieved data to the generator input. We combine RAG with few-shots prompting by including (i) four few-shots and (ii) the retrieved content in the prompt as context for decoding.

Typically, RAG retrieves data based on a given query (here the generation input) and the vector base from which data is retrieved contains data chunks. We explore various options for the query, the vector base chunks and few-shots selection.

We used Llama-3.1-8B-Instruct (Grattafiori et al., 2024) as the base LLM model. Additionally, we ran experiments with Qwen2.5-7B-Instruct (Team, 2024; Yang et al., 2024) and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023).

Paragraph Retrieval. We first experiment with a simple approach where the additional data is the first paragraph of the Wikipedia article associated with the graph root entity. As previously observed (Lebret et al., 2016), this paragraph contains much of the information contained in Wikidata graphs. Hence, we explore a simple approach where, instead of retrieving labels from the Wikidata knowledge base or searching a vector base for relevant content, we include in the prompt the first paragraph of the Wikipedia page corresponding to the root entity of the input graph. As shown in Table 4 (*BL - Graph + WKP Paragraph*), this approach does not improve results, probably because the provided information is too coarse grained introducing a semantic mismatch between the input graph and the retrieved paragraph[5]. We also experiment with another LLM (Mistral) and with combining paragraph selection with random few-shots but again the results were worse than when including Wikidata labels in the prompt and selecting few shots based on their property overlap with the input graph.

Chunks. We derive text chunks from Russian Wikipedia dump. We consider two types of chunks for the retrieval data base: sentence and graph size level.

In *the sentence level* version, each chunk in the retrieval base consists of a single Wikipedia sentence. The sentences are pre-extracted by processing Wikipedia dump [6] and removing all the Wikipedia-specific pages. The parsing was done using mwparserfromhell [7]. Then, the resulting articles are cleaned and split into sentences using the NLTK sentence segmenter (Bird et al., 2009), resulting in more than 60 million sentences. For each triple in the input graph, we retrieve 3 sentences[8].

In the *graph-based* version, we create chunks to match the size of the input graphs (graph-size-chunks) and for each graph of size k, we retrieve n chunks of size k with $n \in \{3, 7\}$. Details on how these graph-size chunks are created are provided in Appendix C. In essence, we use the character length distribution of the texts for each graph size in the WebNLG training data and then we create text

[5]We use the WikipediaRetriever tool from the langchain library to retrieve the first paragraph of the Wikipedia article matching the root entity and add it to the prompt.

[6]https://dumps.wikimedia.org/ruwiki/ accessed in March 2022

[7]https://github.com/earwig/mwparserfromhell

[8]We experimented with retrieving 1, 3 and 7 sentences per triple, 3 yields the best results.

	Seen	Unseen				
	WebNLG	WebNLG		Kelm		NG_{500}
Model	W_S	W_E	W_C	K_E	K_{E+P}	
$[BL]$ - Graph + WKP paragraph	2.38	3.71	3.35	1.42	1.92	6.64
0-Shot RAG						
[1] - Entity based retrieval (ents)	2.53	3.75	3.54	1.49	2.24	5.85
+ Few Shots						
[2] - [1] + R	18.84 (B)	28.82 (B)	**24.63 (A)**	21.41 (A)	16.60 (B)	19.55 (B)
[3] - [1] + P	**31.03 (A)**	**31.95 (A)**	**24.57 (A)**	19.50 (B)	**17.22 (A)**	**20.54 (A)**
Other LLMs						
[2] using Mistral	16.92	24.45	22.81	16.64	16.34	16.46
[2] using Qwen	16.00	22.47	21.79	15.63	13.59	20.08
Other Queries						
[5] - R + entsep	18.06	28.01	24.00	20.64	16.70	19.71
[6] - R + trps	17.56	26.49	21.78	18.79	15.85	17.43
[7] - R + trpsep	17.71	28.89	21.92	18.75	15.65	17.62
Graph-based Chunks						
[5] + 1tblocks (3)	17.64	28.86	23.36	20.44	16.64	19.34
[5] + graph-size-chunks (3)	18.42	26.54	23.82	20.19	15.96	20.04
[5] + graph-size-chunks (7)	18.33	27.35	23.67	20.99	16.32	19.41

Table 4: **BLEU scores for RAG.** The baseline (BL) is a LLama3.1-based 0-shot retrieval model which retrieves a Wikipedia paragraph based on the root entity of the input graph. For all models, except those with graph-based chunks, the vector base chunks are Wikipedia sentences. For the graph-based chunks the number in brackets indicate the number of retrieved chunks per graph (graph-size-chunks) or per triple in the graph (1tblocks).

chunks that simulate that distribution. This way of simulating the block size distribution allows us to have the average length of the chunks in the vector base close to the average character length of the lexicalisations for graphs of each size while also having some variability in the block lengths.

We also experiment with chunks whose text size matches the average length of the WebNLG texts associated with single fact graphs (1tblocks). For a graph of size k, we then retrieve $n \times k$ triple-sized chunks.

In both cases (sentence- and graph-based chunks), chunks are embedded using the LaBSE multilingual encoder (Feng et al., 2022) and applying binary quantization to reduce the size of the embeddings[9]. These are then indexed using a "flat" binary index from the Faiss library (Douze et al., 2024).

Queries. We experiment with the content and the format of the query using either the entities or the facts contained in the input graph. Specifically, the query consists of the entity pairs present in the input graph triples (ents), the same entity pairs but with underscores removed (entsep), the set of triples present in the input graph where triple subject, predicate and object are separated by a space (trps) and the set of triples with underscores removed (trpsep). Appendix D summarises and illustrates the different query types we experimented with.

Few-shots. By default, we use 4 shots but to assess the impact of these few shots, we also consider a 0-shot variant. Based on the results obtained with prompting (see section 5), we select few shots from the Russian WebNLG train and dev data and compare two selection methods: random (R) and based on the property and size similarity with the input graph (P).

Results. Table 4 shows the results highlighting the impact of the various dimensions we explored. The low results for 0-shot RAG underscore the importance of augmenting the prompt with some few-shots. The results also demonstrate that the most impacting factor is the few-shot selection method. As with prompting, selecting few-shot examples based on their property and size similarity with the input graph yields the best results. From the various query types investigated, simply using the entity pairs present in the graph triples yields the best results. Similarly, sentence size chunks (the default setting) yield better results than graph size ones. Finally, a comparison with Mistral and Qwen

[9]The total size of the original vector base was in total more than 200 gb resulting in high retrieval latency. Binary quantization showed similar results on a small batch with a huge advantage in terms of processing time.

shows an advantage for Llama.

7 Comparing the Three Approaches

We compare the three approaches (fine-tuning, prompting, RAG) using the best model we identified for each approach i.e., augmenting the input with Wikidata labels for the Interno model fine-tuned on the Russian WebNLG data (Interno+Labels in Table 2); Prompting with, in addition to the input graph, (i) Russian few-shots selected based on their similarity in terms of size and properties with the input graph and (ii) entity and property labels retrieved from Wikidata (PRU/+L/D in Table 3); and RAG with the same few-shots, entity based queries and sentence level chunks ([3] in Table 4).

7.1 Automatic Evaluation

Figure 1 shows the BLEU scores (scores for other metrics are shown in Appendix G).

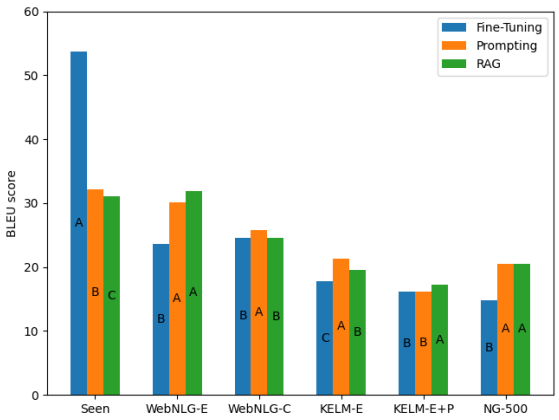

Figure 1: **BLEU Scores for the Best Fine-Tuning, Prompting and RAG models.** Fine-tuning underperforms on unseen data.

Differences between approaches. On seen data, the fine-tuning approach outperforms LLM prompting and RAG by a large margin highlighting the effectiveness of domain adaptation on carefully curated data. However, it systematically underperforms on unseen testsets demonstrating limited generalisation capacity.

Compared to RAG where the added information comes from Wikipedia text, the LLM prompting approach, which extends the input with Russian entity and property labels, has a slight advantage outperforming it or performing on par with it on four out of the five unseen test sets. However, it requires that labels be available in the target language which is a limitation. In contrast, the RAG approach is less constrained simply requiring relevant target language text.

Differences between test sets. Roughly, the scores of the RAG and the prompting models follow a similar trend on the 5 unseen datasets. Notably, scores are highest on W_E and W_C, which is consistent with the fact that the two datasets are available on the web and are likely to have been included in Llama training data. The middle scores on K_E and NG_{500} match the low ratio of labels available for these two datasets (27% and 12% respectively) which in turn often correlates with a low availability of Wikipedia texts in Russian for the corresponding graphs (we observed that entities which have no Russian labels in Wikidata are also often entities for which there is no Wikipedia page in Russian [10]). Finally, the lowest score on K_{E+P} reflects the difficulty of correctly verbalising graphs which contain both unseen entities and unseen properties.

7.2 Human Evaluation

	Seen	Unseen		
		W+K	500	All-U
Faithfulness				
Fine-Tuning	**0.48**	0.37	0.22	0.30
Prompting	0.42	0.47	**0.61**	**0.54**
RAG	0.43	**0.49**	0.50	0.49
Fluency				
Fine-Tuning	**0.45**	0.36	0.29	0.33
Prompting	0.41	0.44	**0.54**	0.49
RAG	0.41	**0.53**	0.50	**0.52**

Table 5: **Human Evaluation Results**. Table 13 in the Appendix shows Fleiss' kappa to estimate inter-annotator agreement.

We also conduct a human evaluation on the three best performing models.

We randomly extract 50 instances from seen WebNLG data, 50 from WebNLG and KELM unseen data, and 50 instances from NG_{500}, totaling 150 samples.

Using the Prolific platform [11] we select 3 annotators who met the following criteria: Russian as their native language, fluency in English, and successful completion of a qualification test assessing their ability to identify additions, omissions, and repetitions in text generated from knowledge graphs.

[10] We looked at all the entities having no Russian labels in the unseen test sets and found that less than 5% of them had a related Wikipedia article in Russian.

[11] https://prolific.com/

To pass the test, annotators were required to provide fully correct responses on at least 10 out of 15 instances.

We show the annotators the original graph in table format as well as 3 texts verbalising it, one for each of the three selected models. Appendix F shows a screenshot of the annotation interface. The order of the models, shown to the annotators, is random and does not contain any information about the models themselves. The annotators are then asked to answer the following four questions about the three texts:

- Please, select the text which is the **closest** to the data shown. Consider both additions and omissions.
- Please, select the text which is the **farthest** from the data shown. Consider both additions and omissions.
- Please, select the text which is the **most fluent**.
- Please, select the text which is the **least fluent**.

Then we treat the answers as a ranking of the models. The closest and the farthest information gives us the ranking for faithfulness, while the most fluent and the least fluent give us the ranking for fluency. We transform each ranking into scores by assigning 3 points to the best model, 1 point to the second, and 0 points to the last. We then accumulate the points by getting their sum and dividing it by the maximum possible score (3 times the number of rankings). Appendix I shows some input/output illustrating the various types of errors made by the three generation methods.

The results of human evaluation are shown in table 5. For fine-tuning, the results confirm the automatic evaluation with the fine-tuned model performing best on seen, and worse on unseen data. On unseen data, prompting is generally judged more faithful but RAG more fluent. We conjecture that the better fluency of RAG stems from the fact that the human written, Wikipedia texts making up the retrieved chunks provide a better context for generation than the stilted WebNLG crowdsourced texts contained in the prompting few-shots. Conversely, the better faithfulness of the prompting approach is likely due to the fact that the labels included in the prompts help improve the verbalisation of unseen entities and properties – this is consistent with the results in Table 3, where augmenting the prompt with Wikidata labels is shown to improve generation performance.

8 Conclusion

Focusing on Russian as a target language, we examine how well different models can convert English-centric knowledge graphs into Russian text on datasets where the input graphs contain unseen entities and/or properties.We compare three common methods for generating text: fine-tuning a pre-trained model on task-specific data, prompting a large language model and Retrieval Augmented Generation. For each method, we design and evaluate multiple variants, selecting the one that performs best.

Across datasets we find that the fine-tuning approach excels on in-domain data but systematically underperforms on unseen data, exposing limited generalisation capacity; that LLM-prompting has a slight advantage over RAG in terms of automatic metrics; and that, based on a human evaluation, RAG produces more fluent, but less faithful text than LLM-prompting. In future work, we plan to examine how these results extend to other languages.

Acknowledgements

We thank the anonymous reviewers for their feed back. This work received government funding managed by the French National Research Agency under France 2030, reference number “ANR-23-IACL-0004.” (AI Chair: "Semantically Consistent LLM Based Text Generation"). Experiments presented in this paper were carried out using the Grid’5000 testbed, supported by a scientific in terest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr). This work was also granted access to the HPC resources of IDRIS under the allocation AD011016561 made by GENCI.

Limitations

We propose to examine how three KG-to-Text generation method perform on unseen data. However, we limit our exploration to graphs stemming from a single knowledge store (Wikidata) and to a single target language (Russian). To overcome this limitation, we plan to extend this work to other languages, considering both High and Low Resource Languages; and to other knowledge or databases with more complex structures such as Rotowire (Wiseman et al., 2017) or MLB (Puduppully et al., 2019).

Ethical statement

Expert feedback is essential for conducting human evaluation. However, prior studies have shown that crowdworkers on platforms such as Amazon Mechanical Turk can be unreliable for open-ended generation tasks (Karpinska et al., 2021) and may not produce trustworthy assessments. To mitigate these issues, we recruited annotators through the Prolific platform, which enabled us to select experts with the relevant level of expertise. All recruited annotators have Russian as their mother tongue and are fluent in English. The Prolific workers were informed that the work they were doing was going to be used for research purposes and were compensated at a rate of £10.14 per hour.

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565, Online. Association for Computational Linguistics.

Oshin Agarwal, Mihir Kale, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2020. Machine translation aided bilingual data-to-text generation and semantic parsing. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 125–130, Dublin, Ireland (Virtual). Association for Computational Linguistics.

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.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020a. 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.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina, editors. 2020b. *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*. Association for Computational Linguistics, Dublin, Ireland (Virtual).

Aleksandr Chuklin, Justin Zhao, and Mihir Kale. 2022. CLSE: Corpus of linguistically significant entities. In *Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 78–96, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Liam Cripwell, Anya Belz, Claire Gardent, Albert Gatt, Claudia Borg, Marthese Borg, John Judge, Michela Lorandi, Anna Nikiforovskaya, and William Soto Martinez. 2023. The 2023 WebNLG shared task on low resource languages. overview and evaluation results (WebNLG 2023). In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 55–66, Prague, Czech Republic. Association for Computational Linguistics.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.

Angela Fan and Claire Gardent. 2020. Multilingual AMR-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2889–2901, Online. Association for Computational Linguistics.

Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2019. Using local knowledge graph construction to scale seq2seq models to multi-document inputs. 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 4177–4187.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic BERT sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from Abstract Meaning

Representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739, San Diego, California. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jie He, Yijun Yang, Wanqiu Long, Deyi Xiong, Victor Gutierrez Basulto, and Jeff Z. Pan. 2025. Evaluating and improving graph to text generation with large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10219–10244, Albuquerque, New Mexico. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Mayank Jobanputra and Vera Demberg. 2024. Team-SaarLST at the GEM'24 data-to-text task: Revisiting symbolic retrieval in the LLM-age. In *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*, pages 92–99, Tokyo, Japan. Association for Computational Linguistics.

Mihir Kale and Abhinav Rastogi. 2020. Template guided text generation for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6505–6520, Online. Association for Computational Linguistics.

Marzena Karpinska, Nader Akoury, and Mohit Iyyer. 2021. The perils of using Mechanical Turk to evaluate open-ended text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1265–1285, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Maxim Kazakov, Julia Preobrazhenskaya, Ivan Bulychev, and Aleksandr Shain. 2023. WebNLG-interno: Utilizing FRED-t5 to address the RDF-to-text problem (WebNLG 2023). In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 67–72, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 388–395.

Nalin Kumar, Saad Obaid Ul Islam, and Ondrej Dusek. 2023. Better translation + split and generate for multilingual RDF-to-text (WebNLG 2023). In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 73–79, Prague, Czech Republic. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 1–9.

Simon Mille, Kaustubh D. Dhole, Saad Mahamood, Laura Perez-Beltrachini, Varun Gangal, Mihir Sanjay Kale, Emiel van Miltenburg, and Sebastian Gehrmann. 2021. Automatic construction of evaluation suites for natural language generation datasets. *CoRR*, abs/2106.09069.

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, Tokyo, Japan. Association for Computational Linguistics.

Anna Nikiforovskaya and Claire Gardent. 2024. Evaluating RDF-to-text generation models for English and Russian on out of domain data. In *Proceedings of the 17th International Natural Language Generation Conference*, pages 134–144, Tokyo, Japan. Association for Computational Linguistics.

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, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Jiaxin Pei, Aparna Ananthasubramaniam, Xingyao Wang, Naitian Zhou, Apostolos Dedeloudis, Jackson Sargent, and David Jurgens. 2022. POTATO: The portable text annotation tool. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 327–337, Abu Dhabi, UAE. Association for Computational Linguistics.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with entity modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2023–2035, Florence, Italy. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. 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 3183–3194, Hong Kong, China. Association for Computational Linguistics.

Phillip Schneider, Manuel Klettner, Elena Simperl, and Florian Matthes. 2024. A comparative analysis of conversational large language models in knowledge-based text generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 358–367, St. Julian's, Malta. Association for Computational Linguistics.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2018. Order-planning neural text generation from structured data. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Xinnuo Xu, Ivan Titov, and Mirella Lapata. 2023. Compositional generalization for data-to-text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9299–9317, Singapore. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Shuzhou Yuan and Michael Faerber. 2023. Evaluating generative models for graph-to-text generation. In *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, pages 1256–1264, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

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

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better AMR-to-text generation. 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 5459–5468, Hong Kong, China. Association for Computational Linguistics.

Dmitry Zmitrovich, Aleksandr Abramov, Andrey Kalmykov, Vitaly Kadulin, Maria Tikhonova, Ekaterina Taktasheva, Danil Astafurov, Mark Baushenko, Artem Snegirev, Tatiana Shavrina, Sergei S. Markov, Vladislav Mikhailov, and Alena Fenogenova. 2024. A family of pretrained transformer language models for Russian. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 507–524, Torino, Italia. ELRA and ICCL.

A Prompting Strategies

We vary the prompt across three dimensions: the input, the generation strategy, and how few shots are selected[12]. We gradually tested and combined the three dimensions, eventually exploring 11 configurations (cf. Table 3).

A.1 Input

We consider different types of input depending on whether the input consists solely of a linearised graph or of the graph augmented with Wikidata extracted information. When enriching the input with Wikidata information, we consider 3 variants: adding property descriptions for all properties present in the graph; adding entity and property descriptions for all entities and properties present in the graph ; and adding target language labels for all properties and entities present in the graph.

Property and entity descriptions are extracted from Wikidata (when available) using the Wikidata *Descriptions* property. To only keep the most informative part of the description, we remove all text within parentheses and any text that appears after the first end-of-clause punctuation (full stop, colon, or semicolon). Similarly, labels are extracted using the Wikidata *Label* property and the resulting dictionary of entity/label pairs is inserted into the input under the key *labels*.

As shown in Table 7, not all Wikidata entities and properties have labels and descriptions and availability varies depending on the target language.

A.2 Generation strategies

We compare two generation strategies: direct and cumulative.

The direct generation strategy (Direct) simply instructs the model to verbalise the input. The cumulative generation strategy (Cumulative) proceeds in multiple steps as follows. First, each individual triple in the input graph is lexicalized separately using the direct approach. The result of each inference call is then collected and passed as input to the LLM with the instruction to combine the triple lexicalisations into a coherent output text [13].

[12]We do not explore 0-Shot as preliminary experiments showed a clear degradation in performance compared to few shots.

[13]When the input graph is of size one, there is no need to continue to the cumulative generation step since there will be only one sentence and the combination step is omitted.

A.3 Few-shot selection strategies

Few shots are retrieved from the Russian dataset (20K graph/text instances). To select the few-shots, we explore 3 strategies: random, size-based and property-based.

Random. We randomly select KG/Text examples making sure that the selected few-shots contain graphs of different size. This approach provides the model with examples of how triples are lexicalised for graphs of different sizes – this helps illustrate the impact of context on the lexicalisation of triples.

Size based. We randomly select KG/Text examples whose KG size is the same as that of the input KG.

Property based. The property-based approach seeks to maximise the property overlap between the input graph and the selected few shots. To this end, all available KG/Text instances are sorted according to their property overlap with the input graph. Top examples are then used as few shots. If there are properties from the input graph that do not appear in the set of selected examples, their (LABSE) encoding is used to search for the most similar properties in WebNLG and the corresponding properties are used to find relevant few shots. The whole process is repeated until the target number of few-shots is reached. Algorithm 1 specifies this selection process in more detail.

Dimensions	Options	Explanation
Input	Graph	The input is the RDF graph
	Prop.	RDF Graph and Wikidata Property Descriptions
	All	RDF Graph, Wikidata Property Descriptions and Entity Descriptions
	P+L	RDF Graph, Wikidata Property Descriptions and Entity Labels
Prompting	Direct	Generate a text verbalising this graph
	Cumulative	Generate a text for each fact in turn then combine the resulting texts into one (two consecutive prompts).
Few-Shot Selection	Random	random examples
	Size-Based	Examples whose graph size is the same as that of the input graph
	Overlap	Examples whose graph maximise property overlap, size and semantic similarity

Table 6: Short Description of the Prompting Options

Algorithm 1 Property-Based Few-Shot Selection

Require: G ▷ Input graph
Require: P_G ▷ Set of properties from G
Require: D ▷ WebNLG dataset
Require: k ▷ Number of few-shots
Require: I ▷ FAISS index of encoded properties
Ensure: FS ▷ Set of few-shot examples

Initialize $FS \leftarrow \emptyset$
Initialize $P \leftarrow P_G$
while $|S| < k$ **do**
  if $|P| = 0$ **then**
    $P \leftarrow P_G$
  end if
  Sort D according to:
    1. Number of overlapping properties with P (descending order)
    2. Size similarity to G (descending order from graphs of size $|G|$ to largest graph size in D)
  if graphs in D overlap with P **then**
    Select the top example e from D
    $S \leftarrow S \cup \{e\}$
    $P \leftarrow P \setminus \mathrm{P}(e)$
  else
    Encode P using LaBSE
    Query FAISS for similar properties P'
    $P \leftarrow P'$
  end if
end while
return S

Dataset	Descs	Labels
WebNLG Testset	35%	46%
KELM-E	19%	27%
KELM-E+P	42%	59%
WebNLG-E	34%	41%
WebNLG-C	34%	43%
Test-500	11%	12%

Table 7: Availability of properties and descriptions in Russian across datasets.

A.4 Example Prompts

An example prompt for direct generation is provided in the listing 1. Example prompts for teo steps of cumulative generation are provided on the listings 2 and 3. Finally, an example prompt with context information (RAG) is provided in listing 4.

```
<|begin_of_text|><|start_header_id|>user<|end_header_id|>

{
    "labels": {
        "country": "государство",
        "United States": " США",
        "foundingDate": "дата основания, создания, возникновения"
    },
    "data": [
        {
            "id": 0,
            "subject": "AmeriGas",
            "property": "country",
            "object": "United States"
        },
        {
            "id": 1,
            "subject": "AmeriGas",
            "property": "foundingDate",
            "object": "1959-01-01"
        }
    ]
}

Generate a lexicalisation of all the 2 triples in Russian.
When necessary, generate text from other languages in their original script.
Include all the information from the triples regardless of their type or relevance.
<|eot_id|><|start_header_id|>assistant<|end_header_id|>

{
    "full-text": "AmeriGas была основана в Соединенных Штатах в 1959-01-01."
}
<|eot_id|><|start_header_id|>user<|end_header_id|>

{
    "labels": {
        "formationDate": "дата основания, создания, возникновения",
        "country": "государство",
        "United States": " США"
    }",
    "data": [
        {
            "id": 0,
            "subject": "Western Goals Foundation",
            "property": "formationDate",
            "object": "01 January 1979"
        },
        {
            "id": 1,
            "subject": "Western Goals Foundation",
            "property": "country",
            "object": "United States"
        }
    ]
}

Generate a lexicalisation of all the 2 triples in Russian.
When necessary, generate text from other languages in their original script.
Include all the information from the triples regardless of their type or relevance.
<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

Listing 1: Example of Direct generation using Overlap One-Shot with labels information in Russian

```
<|begin_of_text|><|start_header_id|>user<|end_header_id|>

{
    "data": [
        {
            "id": 0,
            "subject": "AmeriGas",
            "property": "foundingDate",
            "object": "1959-01-01"
        }
    ]
}

Generate a lexicalisation of all the 1 triples in Russian.
When necessary, generate text from other languages in their original script.
Include all the information from the triples regardless of their type or relevance.
<|eot_id|><|start_header_id|>assistant<|end_header_id|>

{
    "full-text": "AmeriGas была основана в 1959-01-01."
}
<|eot_id|><|start_header_id|>user<|end_header_id|>

{
    "data": [
        {
            "id": 0,
            "subject": "Western Goals Foundation",
            "property": "formationDate",
            "object": "01 January 1979"
        }
    ]
}

Generate a lexicalisation of all the 1 triples in Russian.
When necessary, generate text from other languages in their original script.
Include all the information from the triples regardless of their type or relevance.
<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

Listing 2: Example of a Cumulative generation (Step 1, getting a verbalisation for a single triple) for Russian, using Overlap One-Shot

```
<|begin_of_text|><|start_header_id|>user<|end_header_id|>

{
    "sentences": [
        "AmeriGas была основана в 1959-01-01.",
        "AmeriGas находится в США."
    ]
}

Generate a combined paragraph of all the 2 sentences in Russian.
Include all the information from the sentences regardless of their type or relevance.
<|eot_id|><|start_header_id|>assistant<|end_header_id|>

{
    "full-text": "AmeriGas была основана в Соединенных Штатах в 1959-01-01."
}
<|eot_id|><|start_header_id|>user<|end_header_id|>

{
    "sentences": [
        "Western Goals Foundation была основана 1 января 1979.",
        "Western Goals Foundation находится в США."
    ]
}

Generate a combined paragraph of all the 2 sentences in Russian.
Include all the information from the sentences regardless of their type or relevance.
<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

Listing 3: Example of a Cumulative generation (Step 2, each fact verbalisation aggregation) using Overlap One-Shot Russian

```
<|begin_of_text|><|start_header_id|>user<|end_header_id|>

{
    "context": "Пилот: Алан Шепард. Шпеер, Альберт. 18 ноября 1923 – родился американский астронавт Алан
Бартлет Шепард.",
    "data": [
        {
            "id": 0,
            "subject": "Alan Shepard",
            "property": "occupation",
            "object": "Test pilot"
        },
        {
            "id": 1,
            "subject": "Alan Shepard",
            "property": "birthPlace",
            "object": "New Hampshire"
        },
        {
            "id": 2,
            "subject": "Alan Shepard",
            "property": "birthDate",
            "object": "1923-11-18"
        }
    ],
}

Generate a lexicalisation of all the 3 triples in Russian.
When necessary, generate text from other languages in their original script.
Include all the information from the triples regardless of their type or relevance.
<|eot_id|><|start_header_id|>assistant<|end_header_id|>

{
    "full-text": "Алан Шепард был летчиком-испытателем, который родился в Нью-Гэмпшире 18 ноября 1923 года."
}
<|eot_id|><|start_header_id|>user<|end_header_id|>

{
    "context": "Не Хайшэн () (13 октября 1964) – китайский космонавт (тайконавт). Военный летчик.",
    "data": [
        {
            "id": 0,
            "subject": "Nie Haisheng",
            "property": "birthDate",
            "object": "1964-10-13"
        },
        {
            "id": 1,
            "subject": "Nie Haisheng",
            "property": "occupation",
            "object": "Fighter pilot"
        }
    ],

}

Generate a lexicalisation of all the 2 triples in Russian.
When necessary, generate text from other languages in their original script.
Include all the information from the triples regardless of their type or relevance.
<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

Listing 4: Example of using context information (specifically, entities based RAG with retrieval of one sentence per triple) using Overlap One-Shot in Russian

B Wikidata SparQL Queries

Figures 2 and 3 show example SparQL queries for retrieving Wikidata Russian labels of entities and properties.

```
SELECT ?item ?itemLabel WHERE {
    {
        ?item skos:altLabel "Vasilis Pliatsikas"@en .
    } UNION {
        ?item skos:prefLabel "Vasilis Pliatsikas"@en .
    }
    SERVICE wikibase:label {
        bd:serviceParam
        wikibase:language "ru"
    }
}
```

Figure 2: SparQL query to get labels of entities

```
SELECT ?prop ?propLabel WHERE {
    ?prop rdfs:label "area"@en .
    ?prop wikibase:directClaim ?p .

    SERVICE wikibase:label { bd:serviceParam wikibase:language "ru". }
}
```

Figure 3: SparQL query to get labels of properties

C Creating Graph-Size Chunks

To create graph-size chunks, we first compute the character length distribution (see Figure 4) of the texts for each graph size (1 to 10 triples) in the graph/text WebNLG data. Then we create text chunks that simulate that distribution using the following algorithm. As we merge the sentences extracted from Wikipedia into larger blocks in terms of size based on character number (as shown in the graphs it has a more pronounced single peak), we try to simulate the character number distribution for a specific graph size, assuming it to be a normal distribution with a mean being the same as a centre of a peak. More precisely, to divide a Wikipedia text into blocks for graph size n we use the following algorithm. Let's say we have started a block b and we decide whether we add the next sentence s to it or not. We then draw two random variables:

$$x_1 \sim U(0, f_{\mathcal{N}(m,\sigma)}(|b|))$$

and

$$x_2 \sim U(0, f_{\mathcal{N}(m,\sigma)}(|b+s|))$$

where m is the average character length of the lexicalisations for graphs of the required size, σ refers to a variance we want to have, f is a probability density function and U is a uniform distribution, while $\mathcal{N}$ is a normal one. Then, if $x_1 < x_2$ we add the sentence s to our current block. Otherwise, we start a new block with the sentence s.

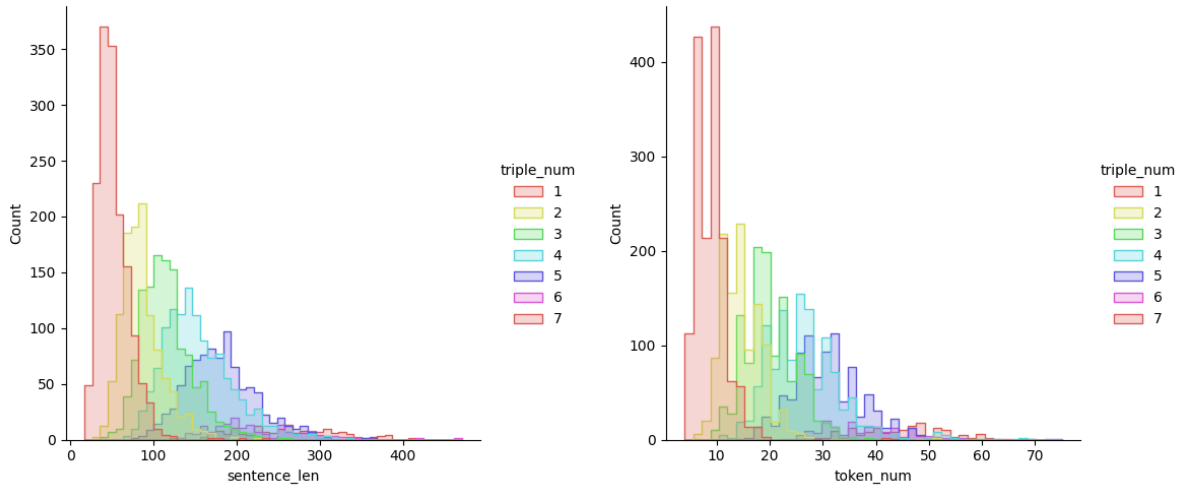

Figure 4: Distribution of sentence length for each graph size in number of symbols (left) and number of tokens (right)

D Query variants for the RAG experiments

In table 8 we summarise the options for queries we experimented on in RAG.

Query Name	Description	Example
Entities (ents)	Entity pair from a triple	"William_Anders Fighter_pilot".
Processed entities (entsep)	Entities pairs from a triple with extra symbols (like underscores) removed	"William Anders Fighter pilot"
Triples (trps)	Triple subject, predicate and object separated by spaces	"William_Anders occupation Fighter_pilot"
Processed triples (trpsep)	Triple subject, predicate and object separated by spaces with underscores removed.	"William Anders occupation Fighter pilot"
Graph entities (unionentsep)	Linearised graph with repeated entities deduplicated	"William Anders occupation Fighter pilot country_of_citizenship United_States "

Table 8: Query variants

E Graph example

In this section we provide an example of a graph which may give advantage to the cumulative generation approach in our prompting variants. The graph is taken directly from the NG_{500} test set and is shown in fig. 5.


```
[
    {
        "subject": "Lufthansa",
        "property": "airline accounting code",
        "object": "220"
    },
    {
        "subject": "Lufthansa",
        "property": "country",
        "object": "Germany"
    },
    {
        "subject": "Lufthansa",
        "property": "callsign of airline",
        "object": "LUFTHANSA"
    },
    {
        "subject": "Lufthansa",
        "property": "official website",
        "object": "https://www.lufthansa.com/de/de/"
    }
]
```


Figure 5: A graph example from NG_{500} test set, with a graph which provides an advantage to cumulative generation approach.

F Human Evaluation GUI

We built an annotation website using potato annotation tool (Pei et al., 2022). An example of GUI can be seen in fig. 6.

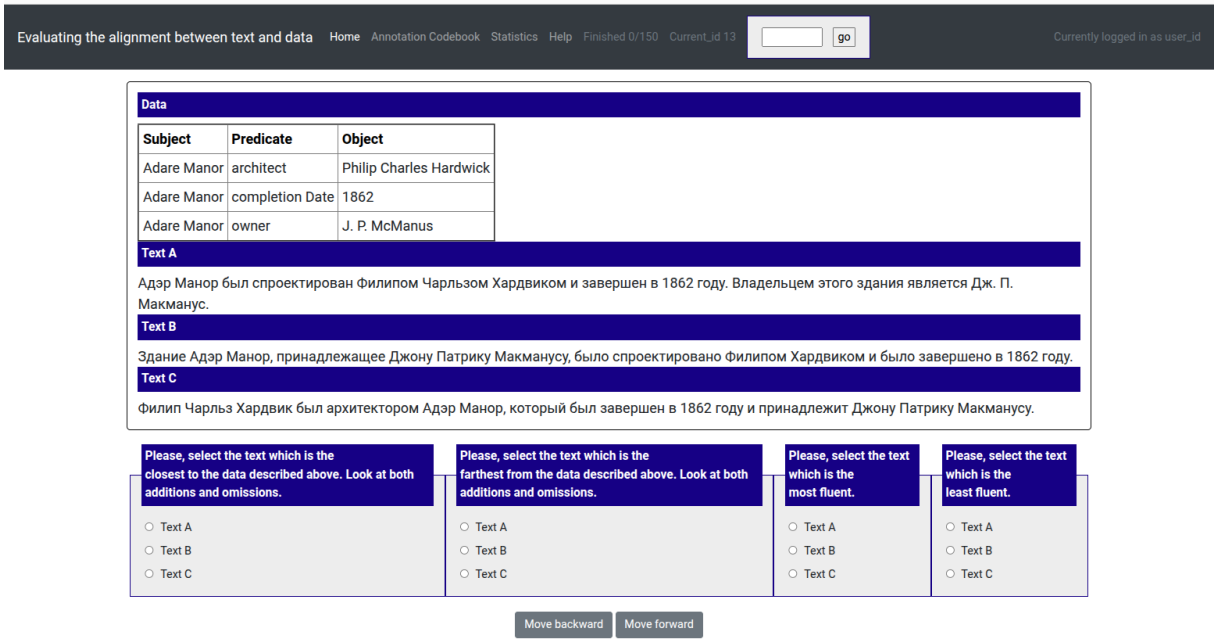

Figure 6: Human annotation GUI example

G Automatic Evaluation Results

We provide complete automatic evaluation results for prompting models, all RAG variations present in the main paper, and some additional experiments that we ran but did not include in the main part of the paper. The results are shown in tables 9,10,11 and 12.

The naming system of the prompting models is the same as the one in the main body of the paper. As for RAG models, we first mention the base LLM, then we precise parameters of the RAG itself. The word 'para' stands for retrieval of the first paragraph based on the root entity. Types of query are the same as in the main body of the paper. Numbers after the type of query are how many blocks are being retrieved with each query. The word 'multi' stands for graph-size chunks. The last part of the name of the model signifies type of few-shots used ('0shots' means no few-shots being used, 'P' stands for few-shots retrieved based on property, 'R' stands for random few-shots, 'S' stands for the few-shots retrieved by similar graph size).

Model	BLEU ↑	chrF++ ↑	TER ↓	BERT P ↑	BERT R ↑	BERT F1 ↑
WebNLG Seen						
PRU/+L/D	32.154	0.572	0.58	0.871	0.868	0.869
PRU/G/D	30.813	0.568	0.594	0.866	0.866	0.865
Llama+para+0shots	2.383	0.202	1.009	0.587	0.706	0.639
Llama+para+P	29.672	0.551	0.619	0.855	0.863	0.858
Mistral+para+P	23.877	0.458	0.715	0.812	0.843	0.825
Llama+para+R	18.092	0.441	0.756	0.809	0.821	0.814
Llama+rag-ents-1+P	30.424	0.566	0.598	0.863	0.865	0.863
Llama+rag-ents-3+R	18.84	0.452	0.732	0.825	0.825	0.824
Qwen+rag-ents-3+R	16.004	0.422	0.795	0.8	0.807	0.802
Mistral+rag-ents-3+R	16.924	0.439	0.764	0.813	0.817	0.814
Llama+rag-ents-3+P	31.033	0.568	0.598	0.864	0.866	0.864
Llama+rag-trps-3+R	17.559	0.442	0.756	0.817	0.818	0.817
Llama+rag-entsep-3+R	18.547	0.452	0.73	0.826	0.824	0.824
Llama+rag-trpsep-3+R	17.712	0.438	0.749	0.817	0.817	0.816
Llama+rag-1tblocks-entsep-3+R	17.635	0.45	0.75	0.82	0.823	0.821
Llama+rag-multi-entsep-3+R	18.421	0.446	0.741	0.821	0.822	0.82
Llama+rag-multi-entsep-7+R	18.328	0.44	0.741	0.819	0.822	0.819
Llama+rag-multi-unionentsep-7+R	18.6	0.448	0.732	0.824	0.824	0.823
Llama+rag-ents-3+0shots	2.534	0.217	1.03	0.59	0.701	0.639
Llama-rag-ents-3+S	19.35	0.459	0.722	0.828	0.828	0.827

Table 9: **Results of LLMs with different prompting methods and RAG on the seen WebNLG test set**

Model	BLEU ↑	chrF++ ↑	TER ↓	BERT P ↑	BERT R ↑	BERT F1 ↑
WebNLG E						
PRU/+L/D	30.088	0.579	0.582	0.882	0.882	0.881
PRU/G/D	29.764	0.584	0.573	0.88	0.883	0.88
Llama+para+0shots	3.712	0.246	5.414	0.604	0.722	0.656
Llama+para+P	29.671	0.578	0.578	0.882	0.882	0.881
Mistral+para+P	21.614	0.509	1.077	0.838	0.859	0.847
Llama+para+R	25.953	0.549	0.612	0.875	0.872	0.872
Llama+rag-ents-1+P	27.54	0.574	0.6	0.875	0.878	0.875
Llama+rag-ents-3+R	28.819	0.567	0.593	0.876	0.873	0.873
Qwen+rag-ents-3+R	22.47	0.482	0.71	0.841	0.843	0.84
Mistral+rag-ents-3+R	24.447	0.518	0.647	0.858	0.857	0.857
Llama+rag-ents-3+P	31.947	0.579	0.557	0.883	0.886	0.884
Llama+rag-trps-3+R	26.488	0.54	0.615	0.875	0.869	0.87
Llama+rag-entsep-3+R	28.014	0.562	0.626	0.873	0.871	0.871
Llama+rag-trpsep-3+R	28.894	0.556	0.605	0.874	0.867	0.869
Llama+rag-1tblocks-entsep-3+R	28.859	0.571	0.603	0.878	0.874	0.875
Llama+rag-multi-entsep-3+R	26.543	0.559	0.636	0.871	0.873	0.871
Llama+rag-multi-entsep-7+R	27.346	0.553	0.607	0.874	0.872	0.872
Llama+rag-multi-unionentsep-7+R	28.914	0.567	0.579	0.877	0.873	0.874
Llama+rag-ents-3+0shots	3.754	0.292	4.674	0.617	0.729	0.666
Llama+rag-ents-3+S	28.44	0.557	0.592	0.874	0.872	0.872
WebNLG C						
PRU/+L/D	25.757	0.491	0.638	0.85	0.847	0.847
PRU/G/D	24.764	0.476	0.653	0.844	0.842	0.842
Llama+para+0shots	3.354	0.224	1.011	0.585	0.714	0.641
Llama+para+P	24.29	0.475	0.674	0.836	0.844	0.838
Mistral+para+P	20.913	0.429	0.746	0.807	0.83	0.816
Llama+para+R	22.849	0.469	0.693	0.831	0.841	0.834
Llama+rag-ents-1+P	24.577	0.477	0.651	0.844	0.842	0.842
Llama+rag-ents-3+R	24.631	0.478	0.657	0.844	0.841	0.841
Qwen+rag-ents-3+R	21.793	0.453	0.711	0.825	0.826	0.824
Mistral+rag-ents-3+R	22.811	0.464	0.678	0.834	0.833	0.832
Llama+rag-ents-3+P	24.568	0.479	0.655	0.844	0.842	0.841
Llama+rag-trps-3+R	21.778	0.453	0.698	0.83	0.832	0.83
Llama+rag-entsep-3+R	24.004	0.468	0.667	0.839	0.838	0.837
Llama+rag-trpsep-3+R	21.918	0.453	0.707	0.828	0.831	0.828
Llama+rag-1tblocks-entsep-3+R	23.356	0.465	0.675	0.837	0.839	0.837
Llama+rag-multi-entsep-3+R	23.823	0.47	0.67	0.84	0.841	0.839
Llama+rag-multi-entsep-7+R	23.668	0.472	0.67	0.84	0.841	0.839
Llama+rag-multi-unionentsep-7+R	23.292	0.463	0.67	0.837	0.836	0.836
Llama+rag-ents-3+0shots	3.535	0.239	1.012	0.59	0.701	0.638
Llama+rag-ents-3+S	23.41	0.47	0.665	0.841	0.839	0.839

Table 10: **Results of LLMs with different prompting methods and RAG on WebNLG-based unseen test sets**

Model	BLEU ↑	chrF++ ↑	TER ↓	BERT P ↑	BERT R ↑	BERT F1 ↑
KELM-E						
PRU/+L/D	21.359	0.512	0.609	0.865	0.866	0.865
PRU/G/D	21.54	0.504	0.608	0.864	0.864	0.864
Llama+para+0shots	1.422	0.129	16.022	0.543	0.697	0.61
Llama+para+P	20.678	0.491	0.757	0.855	0.858	0.856
Mistral+para+P	15.053	0.391	1.925	0.804	0.836	0.818
Llama+para+R	19.943	0.491	0.81	0.85	0.858	0.853
Llama+rag-ents-1+P	19.634	0.485	0.67	0.855	0.857	0.856
Llama+rag-ents-3+R	21.408	0.505	0.632	0.859	0.86	0.859
Qwen+rag-ents-3+R	15.634	0.414	0.875	0.818	0.825	0.821
Mistral+rag-ents-3+R	16.64	0.456	0.78	0.833	0.841	0.837
Llama+rag-ents-3+P	19.502	0.482	0.648	0.858	0.858	0.857
Llama+rag-trps-3+R	18.788	0.481	0.865	0.843	0.851	0.846
Llama+rag-entsep-3+R	20.642	0.483	0.769	0.851	0.853	0.852
Llama+rag-trpsep-3+R	18.754	0.474	0.883	0.84	0.847	0.843
Llama+rag-1tblocks-entsep-3+R	20.441	0.484	0.74	0.848	0.852	0.85
Llama+rag-multi-entsep-3+R	20.186	0.491	0.725	0.85	0.853	0.851
Llama+rag-multi-entsep-7+R	20.987	0.49	0.739	0.853	0.854	0.853
Llama+rag-multi-unionentsep-7+R	20.3	0.489	0.709	0.854	0.854	0.854
Llama+rag-ents-3+0shots	1.488	0.135	14.628	0.544	0.682	0.604
Llama-rag-ents-3+S	20.442	0.496	0.632	0.858	0.857	0.857
KELM-E+P						
PRU/+L/D	16.193	0.46	0.867	0.816	0.835	0.825
PRU/G/D	16.509	0.455	0.823	0.823	0.833	0.828
Llama+para+0shots	1.916	0.184	10.076	0.557	0.679	0.611
Llama+para+P	16.038	0.438	1.062	0.812	0.83	0.82
Mistral+para+P	14.541	0.423	1.377	0.795	0.826	0.809
Llama+para+R	15.726	0.448	0.935	0.813	0.83	0.821
Llama+rag-ents-1+P	16.917	0.456	0.807	0.824	0.834	0.829
Llama+rag-ents-3+R	16.599	0.449	0.866	0.823	0.831	0.826
Qwen+rag-ents-3+R	13.587	0.424	0.981	0.797	0.814	0.805
Mistral+rag-ents-3+R	16.342	0.452	0.843	0.821	0.83	0.825
Llama+rag-ents-3+P	17.218	0.458	0.828	0.825	0.834	0.829
Llama+rag-trps-3+R	15.847	0.441	0.918	0.817	0.827	0.821
Llama+rag-entsep-3+R	16.702	0.455	0.843	0.822	0.831	0.826
Llama+rag-trpsep-3+R	15.653	0.446	0.882	0.817	0.828	0.822
Llama+rag-1tblocks-entsep-3+R	16.64	0.452	0.837	0.822	0.83	0.826
Llama+rag-multi-entsep-3+R	15.959	0.446	0.926	0.819	0.83	0.824
Llama+rag-multi-entsep-7+R	16.322	0.45	0.895	0.821	0.832	0.826
Llama+rag-multi-unionentsep-7+R	16.296	0.45	0.867	0.822	0.832	0.826
Llama+rag-ents-3+0shots	2.242	0.212	7.385	0.574	0.689	0.626
Llama+rag-ents-3+S	16.143	0.454	0.839	0.822	0.832	0.826

Table 11: **Results of LLMs with different prompting methods and RAG on KELM-based test sets**

Model	BLEU ↑	chrF++ ↑	TER ↓	BERT P ↑	BERT R ↑	BERT F1 ↑
NG_{500}						
PRU/+L/D	20.543	0.437	0.723	0.825	0.82	0.822
PRU/G/D	21.51	0.441	0.711	0.824	0.822	0.822
Llama+para+0shots	6.635	0.275	4.865	0.603	0.72	0.655
Llama+para+P	19.641	0.426	1.012	0.802	0.814	0.807
Llama+para+R	18.508	0.414	0.865	0.806	0.81	0.807
Llama+rag-ents-3+R	19.554	0.425	0.746	0.819	0.813	0.815
Qwen+rag-ents-3+R	20.081	0.438	0.84	0.805	0.815	0.81
Mistral+rag-ents-3+R	16.46	0.407	0.806	0.807	0.805	0.805
Llama+rag-ents-3+P	20.543	0.433	0.753	0.817	0.817	0.816
Llama+rag-ents-3+S	19.928	0.421	0.747	0.818	0.813	0.815
Llama+rag-trps-3+R	17.434	0.398	0.792	0.81	0.803	0.806
Llama+rag-entsep-3+R	19.707	0.424	0.736	0.822	0.814	0.817
Llama+rag-trpsep-3+R	17.617	0.394	0.873	0.813	0.803	0.807
Llama+rag-1tblocks-entsep-3+R	19.336	0.421	0.766	0.819	0.811	0.814
Llama+rag-multi-entsep-3+R	20.035	0.428	0.782	0.819	0.815	0.817
Llama+rag-multi-entsep-7+R	19.409	0.418	0.782	0.818	0.814	0.815
Llama+rag-multi-unionentsep-7+R	18.716	0.409	0.813	0.815	0.81	0.812
Llama+rag-ents-3+0shots	5.848	0.264	4.778	0.598	0.709	0.648

Table 12: **Results of LLMs with different prompting methods and RAG on NG_{500} test set**

H Human Evaluation: agreement

In table 13, we present Fleiss' kappa to estimate inter-annotator agreement between our three annotators.

	All	**By source**		
		Seen	W+K	500
closest	0.22	0.25	0.17	0.19
farthest	0.23	0.28	0.13	0.18
most fluent	0.17	0.15	0.18	0.16
least fluent	0.16	0.08	0.24	0.19

Table 13: **Fleiss' kappa**. Estimating inter-annotator agreement on each question on all of data, as well as on each part of it by source.

I Comparing the Three Approaches: case study

In table 14 we provide an example, where our annotators have deemed the RAG model as the most fluent, Prompting model as the closest to the data and our Fine-Tuning model (Interno+Labels) as both the farthest produced text from the data and the least fluent. The example is taken from a NG_{500} dataset. To make the analysis of the models easier, we provide translations of the texts to English, trying to keep the same mistakes when possible. We also use a system of different underline styles to show which parts of the texts refer to which fact (triple of the graph).

The simple underlined text signifies that the lexicalisation of the fact is correct. The *emphasized orange underlined text* signifies that the lexicalisation partly describes the fact. The *emphasized orange text* shows unnecessary repetition in the text, and the **bold blue text** shows information not present in the data. Different underline styles (like, one line, dot line, wave line) highlight text fragments which are supposed to be verbalisations of the corresponding triples.

In the example, we can see that the Prompting method stays closest to the described data, however, it mostly states the facts without connecting them properly. This results in a loss of perceived fluency. RAG model verbalises all the facts correctly, however, it additionally states a fact that can be derived from the chemical formula. It is correct, but it can be considered a slight difference from the data. Finally, we can see that the Fine-Tuning model did not verbalize all of the facts, and overall produced a poor text with repetitions and information not present in the data.

Graph	[["guanethidine", "has use", "medication"], ["guanethidine", "ATC code", "S01EX01"], ["guanethidine", "canonical SMILES", "C1CCCN(CCC1)CCN=C(N)N"], ["guanethidine", "InChI", "InChI=1S/C10H22N4/c11-10(12)13-6-9-14-7-4-2-1-3-5-8-14/h1-9H2,(H4,11,12,13)"], ["guanethidine", "chemical formula", "$C_{10}H_{22}N_4$"]]
RAG	Гуанетидин представляет собой препарат, который используется в качестве лекарства. Его ATC-код - S01EX01. Химическая формула гуанетидина - C10H22N4. *В его структуре присутствуют атомы углерода, водорода, азота.* Вещество имеет канонический SMILES-код C1CCCN(CCC1)CCN=C(N)N и InChI-код InChI=1S/C10H22N4/c11-10(12)13-6-9-14-7-4-2-1-3-5-8-14/h1-9H2,(H4,11,12,13).
Translation	Guanethidine is a product that is used as a medicine. Its ATC code is S01EX01. The chemical formula of guanethidine is C10H22N4. *Its structure contains carbon, hydrogen, and nitrogen atoms.* The substance has the canonical SMILES code C1CCCN(CCC1)CCN=C(N)N and InChI code InChI=1S/C10H22N4/c11-10(12)13-6-9-14-7-4-2-1-3-5-8-14/h1-9H2,(H4,11,12,13).
Prompting	Гуанетидин является лекарственным средством, имеет ATC код S01EX01, химическую формулу $C_{10}H_{22}N_4$, а также SMILES-код C1CCCN(CCC1)CCN=C(N)N и InChI-код InChI=1S/C10H22N4/c11-10(12)13-6-9-14-7-4-2-1-3-5-8-14/h1-9H2,(H4,11,12,13).
Translation	Guanethidine is a medication, has ATC code S01EX01, chemical formula $C_{10}H_{22}N_4$, and also SMILES code C1CCCN(CCC1)CCN=C(N)N and InChI code InChI=1S/C10H22N4/c11-10(12)13-6-9-14-7-4-2-1-3-5-8-14/h1-9H2,(H4,11,12,13).
Fine-Tuning	*1S/C10H22N4* - лекарственное средство, которое имеет *C1CCCN(CCC1)CCN=C(N)N* и canonical smiles C1CCCN(CCC1)CCN=C(N)N. Оно имеет ATC код S01EX01 и **может быть использовано как для внутреннего, так и для наружного применения.**
Translation	*1S/C10H22N4* is a medication that has *C1CCCN(CCC1)CCN=C(N)N* and canonical smiles C1CCCN(CCC1)CCN=C(N)N. It has ATC code S01EX01 and **can be used for both internal and external use.**

Table 14: An example of a graph and three systems' outputs, where our annotators have marked Prompting as the closest model to the data, Fine-Tuning as the farthest text from the data, RAG as the most fluent text and Fine-Tuning as the least fluent text. The simple underlined text signifies that the lexicalisation of the fact is correct. The emphasized orange underlined text signifies that the lexicalisation partly describes the fact. The emphasized orange text shows unnecessary repetition in the text, and the bold blue text shows information not present in the data.

J Hyperparameters

For all LLMs we set `max_new_tokens` to 1200, `temperature` to 0.7 and `top_p` value equal to 0.7. We run all our experiments on a single GPU Nvidia A40 with 45GB of memory.