Lattice @MultiGEC-2025: A Spitful Multilingual Language Error Correction System Using LLaMA

Olga Seminck[1], **Yoann Dupont**[1], **Mathieu Dehouck**[1], **Qi Wang**[1], **Noé Durandard**[1], **Margo Novikov**[1]

[1]LaTTiCe UMR8094 : CNRS, ENS-PSL, Sorbonne-Nouvelle, Paris, France
olga.seminck@cnrs.fr, yoann.dupont@sorbonne-nouvelle.fr, mathieu.dehouck@cnrs.fr, qi.wang@cnrs.fr, noe.durandard@psl.eu, margosha.novikova@gmail.com

Abstract

This paper reports on our submission to the NLP4CALL shared task on Multilingual Grammatical Error Correction (MultiGEC-2025) (Masciolini et al., 2025). We developed two approaches: fine-tuning a large language model, LLaMA 3.0 (8B), for each MultiGEC corpus, and a pipeline based on the encoder-based language model XLM-RoBERTa. During development, the first method significantly outperformed the second, except for languages that are poorly supported by LLaMA 3.0 and have limited MultiGEC training data. Therefore, our official results for the shared task were produced using the neural network system for Slovenian, while fine-tuned LLaMA models were used for the eleven other languages. In this paper, we first introduce the shared task and its data. Next, we present our two approaches, as well as a method to detect cycles in the LLaMA output. We also discuss a number of hurdles encountered while working on the shared task.

1 Introduction

South American camelids are infamous for spitting at each other and at people's faces. Working on the MultiGEC-2025 shared task on grammatical error correction, we realized that LLaMA 3.0 is no different.

Grammatical Error Correction (GEC) is a fundamental task in Natural Language Processing (NLP) for bureautics and educational settings, aimed at automatically identifying and correcting grammatical errors in written texts. As a helping tool in second language acquisition (Volodina et al., 2023), it is essential for addressing diverse learning needs and backgrounds (Loem et al., 2023) that GEC be able to handle various modes of correction, such as minimal and fluency edits,. Minimal edit correction focuses on addressing grammatical errors while preserving the original form and structure of the text, whereas fluency correction involves rewriting texts to enhance idiomaticity and achieve greater naturalness (Davis et al., 2024). The errors to identify are not limited to grammatical errors; other types such as orthographical, syntactical and lexical errors need also be considered.

Research in GEC has advanced significantly over the past decades, from rule-based methods (Sidorov et al., 2013) to statistical approaches (Yuan and Felice, 2013), followed by neural network models (Bryant et al., 2023), and most recently, large language models (LLMs), such as OpenAI's GPT and Meta's LLaMA LLMs (Davis et al., 2024).

The objective of the NLP4CALL shared task, MultiGEC-2025 (Masciolini et al., 2025), is to perform grammatical error correction (GEC) on 12 languages: Czech, English, Estonian, German, Greek, Icelandic, Italian, Latvian, Russian, Slovene, Swedish, and Ukrainian. The shared task requires rewriting texts produced by language learners to make them either grammatically correct (minimal edits) or both grammatically correct and idiomatic (fluency edits) (Table 1).

In this paper, we present the systems submitted by our team, Lattice, to the MultiGEC-2025 shared task, which was hosted as part of the 14th Workshop on Natural Language Pro-

Olga Seminck, Yoann Dupont, Mathieu Dehouck, Qi Wang, Noé Durandard and Margo Novikov. Lattice @MultiGEC-2025: A Spitful Multilingual Language Error Correction System Using LLaMA. *Proceedings of the 14th Workshop on Natural Language Processing for Computer Assisted Language Learning (NLP4CALL 2025)*. University of Tartu Library. 34–41.

cessing for Computer-Assisted Language Learning (NLP4CALL). The structure of this paper is as follows: first, we provide an overview of the dataset introduced by the organizers. Next, we describe the two methods we developed: one based on fine-tuning the LLaMA 3.0 model (Touvron et al., 2023), and the other based on the XLM-RoBERTa language model (Conneau et al., 2020), followed by a method to detect and remove cycles in the LLaMa output that we used to enhance the results. Finally, we present an analysis of our system's results in relation to the particularities of the 17 different corpora included in the shared task's data.

Input	My mother became very sad, no food. But my sister better five months later.
Minimal Correction	My mother became very sad, and ate no food. But my sister felt better five months later.
Fluency Correction	My mother was very distressed and refused to eat. Luckily, my sister recovered five months later.

Table 1: An example of an input text with reference corrections (minimal and fluency edits).

2 Data

Table 2 presents detailed statistics of the corpus, including the number of essays, word counts, and sentence counts, calculated using the syntok tokenizer provided by the MultiGEC organizers. These statistics focus exclusively on the original written essays (i.e., excluding rewritten essays produced by our systems). The data highlights the structural diversity across the datasets, with essay lengths varying significantly depending on the source language of each dataset.

On average, each essay contains 277 tokens. Notably, Icelandic essays are considerably longer, with the IceEC dataset averaging 1,004 tokens per essay and the IceL2EC dataset averaging 818 tokens. Similarly, Slovene essays in the Solar-Eval dataset average 642 tokens per essay. In contrast, the Russian dataset has a significantly lower average, with essays containing only 38 tokens per essay.

Most datasets follow the traditional split of approximately 80% for training, 10% for development, and 10% for testing. However, there are exceptions. The Slovene dataset is relatively small, consisting of only 109 essays. Its splits are notably unbalanced, with 10 essays in the training set, 50 in the development set, and 49 in the test set. In contrast, the Russian dataset is split more evenly, with 42% allocated for training, 33% for development, and 25% for testing.

3 Methods

3.1 Baseline

The MultiGEC organizers offer a one-shot multilingual baseline leveraging the LLaMA 3.1 8B Instruct model. In this approach, a single example in English is incorporated into the prompt, addressing binary scenarios that focus on either minimal edits or fluency edits.

3.2 Fine-tuned LLaMA 3.0 8B

The methodology applied by our team was *peft* (parameter-efficient fine-tuning) with 4-bit NormalFloat quantization using QLora (Dettmers et al., 2024), a method based on *Low-Rank Adaptation* (LoRA) (Biderman et al., 2024), for fine-tuning the LLaMA 8B 3.0 model (Touvron et al., 2023). We choose peft due to its higher efficiency in terms of computational requirements and its ability to prevent model collapse and catastrophic forgetting [1]. We utilized a single RTX 3090 GPU with 24GB of RAM for training and testing. The process consumed approximately 97% of the available memory, indicating that experimenting with a heavier model would likely be infeasible given our current infrastructure.

During the development phase, we observed that fine-tuning a model on each training set and then applying it to the corresponding development set led to improved performance compared to using a single model per language. We thus decided to fine-tune one model per corpus, resulting in 17 models in total. During the test phase, training data consisted of the concatenation of the MultiGEC training and development sets, as we noticed that corpora with more training data tended to achieve higher performance.

During the development phase, we also observed that essays lacking a strong punctuation

[1] https://ai.meta.com/blog/how-to-fine-tune-llms-peft-dataset-curation/

Lang.	Source	Split	#Essays	#Sents.	#Tokens
Czech	NatForm	train	227	3,245	44,261
		dev	88	1,537	22,206
		test	76	1,433	19,962
Czech	NatWebInf	train	3,620	6,463	87,345
		dev	1,291	2,270	31,118
		test	1,256	2,059	26,963
Czech	SecLearn	train	2,057	27,741	331,953
		dev	173	2,608	32,106
		test	177	2,710	35,264
Czech	Romani	train	3,247	18,198	280,268
		dev	179	900	14,616
		test	173	967	15,706
English	Write & Improve	train	4,040	37,341	680,405
		dev	506	4,307	89,132
		test	504	4,911	93,419
Estonian	EIC	train	206	2,849	33,923
		dev	26	366	4,491
		test	26	385	4,344
Estonian	EKIL2	train	1,202	14,400	189,162
		dev	150	1,853	24,546
		test	151	1,676	23,103
German	Merlin	train	827	8,455	117,345
		dev	103	1,102	15,762
		test	103	1,029	13,361
Greek	GLCII	train	1,031	12,167	207,606
		dev	129	1,538	26,385
		test	129	1,525	24,640
Icelandic	IceEC	train	140	7,146	141,439
		dev	18	784	16,028
		test	18	905	19,178
Icelandic	IceL2EC	train	155	5,470	124,750
		dev	19	741	18,899
		test	19	595	14,329
Italian	Merlin	train	651	6,620	83,419
		dev	81	818	10,704
		test	81	845	10,562
Latvian	LaVA	train	813	17,254	148,701
		dev	101	2228	18,514
		test	101	2,091	17,995
Russian	RULEC-GEC	train	2,539	5,191	90,424
		dev	1,969	2,688	45,260
		test	1,535	5,321	92,337
Slovene	Solar-Eval	train	10	253	5,062
		dev	50	1,672	31,365
		test	49	1,775	33,515
Swedish	SweLL_gold	train	402	6,294	120,433
		dev	50	724	13,232
		test	50	653	12,066
Ukrainian	UA-GEC	train	1,706	29,429	460,385
		dev	87	1,318	23,953
		test	79	1,089	20,030

Table 2: MultiGEC data statistics (original files using the syntok tokenizer).

marker at the end often resulted in excessively long outputs. To address this, we added a stop token ("$$$") to the end of each essay. After generation, these stop tokens were removed.

To formalize the roles of the prompt, input, and correct output, we transformed the data into a .json format. The prompt it the same for all 17 corpora and is taken from the original provided base-

line: *"You are a grammatical error correction tool. Your task is to correct the grammaticality and spelling of the input essay written by a learner. Return only the corrected text and nothing more."*

The tokens per essay were counted using the LLaMA tokenizer[2]. For each corpus, the maximum number of tokens was used to determine the maximum generation length, which was set to this number plus 15%. For example, for Italian, the maximum token length was 478, so this parameter was set to 550. Keeping this number as low as possible is important because setting it too high significantly slows down the prediction process.

For all languages, we used batches of 10 essays and set the gradient accumulation parameter to 4 steps. The number of optimization steps is provided in Table 3. Initially, we set the number of optimization steps to 500. However, after training about half of the models, we realized that we did not have enough time left before the system submission deadline, so we trained the remaining models with fewer optimization steps.

We did not conduct quantitative research on the relationship between the number of optimization steps and model performance. Therefore, it is possible that similar performance could be achieved with fewer optimization steps or that better performance could be obtained with more optimization steps.

3.2.1 Detection of Cycles in LLaMA's Outputs

Despite using a stop token ("$$$") to help the model interrupt the generation process, the model still occasionally loops and repeats the same sentence (or a few sentences) until it reaches the allotted number of tokens for the current essay.

In order to mitigate this undesired behaviour, we passed the outputs of the LLaMA model to an ad-hoc repetition detector which works as follows: Given a string of characters s_i with i ranging from 0 to l, the length of the string. For each character n-gram r ($n = 15$ in this case), we get the sorted list of indices at which r appears in s. From this list, we compute the distance between each pair of consecutive occurrences of r. Eventually, if there are at least 20 occurrences of r with 10 or more pairs of the same distance, especially toward the end of the essay, we flag the essay.

In theory, a model could recover from a cycle since its internal state evolves during the generation process, and it could also experience very long cycles. However, in practice, detecting 10 or more similarly spaced occurrences of 15 characters at the end of an essay was sufficient for capturing LLaMA's loops.

This tool was used for both diagnosis and intervention. When only a few essays are flagged for loops, we address them with a simple rule. If the loop is a well-formed sentence (i.e., it starts with an uppercase letter and ends with punctuation), we cut the essay after the first occurrence of the loop. If the beginning of the loop can be found in the original essay, we append the end of the original essay to our correction. If the loop is not a well-formed sentence, we cut at the end of the last well-formed sentence before the loop and append the rest of the original essay. This approach helps avoid losing too much of the essay if the loop occurs early in the text.

For example, in the Czech NatForm corpus, out of the 646 essays, two had loops (30 repetitions of 115 characters for one, and 41 repetitions of 42 characters for the other). The 30 repetitions correspond to the sequence "*Na druhém boku je zástrčka na sluchátka. Na vrchu mobilu je zástrčka na nabíjení, USB kabel a tlačítko na vypnutí.*" The detected r is actually ". *Na druhém bok*", since it is present in the original essay, we remove the repeating section and replace it with the end of the original essay.

Fined-tuned Llama did not generate looping text on most corpora. The Czech NatWebInf had one problematic essay (out of 1256) due to a long string of dashes "-". There was one problematic essay in the Greek GLCII corpus (out of 481), and one in the Ukrainian UA_GEC corpus (out of 456). However, from the 49 Slovene Solar-Eval essays, 11 were problematic (22.44%) and virtually all Icelandic essays were problematic as well. As a result, we decided to use use the XLM-Roberta detect and correct model (described below) for Slovene and to simply return the original, untouched texts for Icelandic.

[2]Note that this is a different tokenizer from the syntok tokenizer used to count the number of tokens in Table 2. The syntok tokenizer is used to separate words and punctuation symbols in order to compute the various scores of a proposed correction, while LLaMA's tokenizer is used internally by the language model for vectorizing its inputs in order to deal with rare or out of vocabulary words.

Lang.	Source	#Optimization Steps
Czech	NatForm	150
Czech	NatWebInf	500
Czech	SecLearn	200
Czech	Romani	150
English	Write & Improve	200
Estonian	EIC	150
Estonian	EKIL2	150
German	Merlin	500
Greek	GLCII	150
Icelandic	IceEC	500
Icelandic	IceL2EC	500
Italian	Merlin	500
Latvian	LaVA	500
Russian	RULEC-GEC	150
Slovene	Solar-Eval	50
Swedish	SweLL_gold	150
Ukrainian	UA-GEC	200

Table 3: The number of optimization steps per corpus during the fine-tuning of the LLaMA models.

3.3 Detect and Correct Errors with XLM-Roberta

This alternative approach models the task as a two-stage prediction. The first stage involves detecting errors in the source data as a token labeling task. The second stage revolves around using a masked language model to generate a token as a replacement of a token labeled as an error in the first stage. In both stages, we used XLM-RoBERTa encoder-based language model (Conneau et al., 2020).

For the first stage, we need to create a labeled corpus from the source and gold essays. We apply a variant of the Needleman-Wunsch algorithm used to compute Levenshtein distance. In the classic algorithm, every error is given a weight of 1, which could cause some misalignment when there is a string of errors. To prevent misalignment of tokens, we give a (usually) lower weight to substitution edits to favor them instead of deletions and insertions. The actual weight for substitutions is computed using the `ratio` function provided by the python library called "Levenshtein"[3].

Once we have collected all the edit operations to transform an original sentence into its reference counterpart, we use these edits to create labels on the tokenized original sentence. Deletions (a token that is present in the original sentence but not the reference) are labeled "-". Insertions (a token from reference that should be added to the sentence) are ignored as they cannot be processed easily as part of a token classification task. Substitutions (when a token was in original sentence was partially aligned with a token in its reference counterpart) can be mapped to labels with a varying degree of granularity. We tried two variants: a coarse-grain and a fine-grain label scheme. In the fine-grain label scheme, we computed labels given predefined error types. We handled three casing modifications: to lower case, to upper case and to title case. We also modeled suffix modification for only the last letter, the tag is the letter to use to correct the token. Errors that did not fit into any previous case were given a generic error label marked as "<mask>". In the coarse-grain label scheme, only the "<mask>" label is used.

The token classification model is trained by fine-tuning XLM-RoBERTa embeddings with the flair library (Akbik et al., 2019).

For prediction, unlabeled essays are first tokenized using the syntok library[4]. The fine-tuned token classification model is then applied to label the data using the flair library. When the label represents a predefined correction to apply (to lower/upper case, substitute last letter, etc.), it is directly applied to the token. When a token is labeled with the generic error label, we apply XML-Roberta as a masked language model (MLM) to output the best token given the context of the sentence. The only optimization we tried is providing a threshold for the probability of the token that the MLM predicts. We used a threshold probability of 0.75 to apply a change. That is, the probability of

[3]The library is available at the following URL: `https://github.com/rapidfuzz/Levenshtein`

[4]https://github.com/fnl/syntok/

the predicted token by XLM-roberta has to be at least 0.75 or else the token is left unchanged.

In the end, labels other than "<mask>" and deletions were scarce and not well recognized by trained models. The final flair model used the coarse-grain label scheme.

4 Results

Systems are evaluated on automatic metrics categorized into two groups : reference-based metrics, including GLEU, Precision, Recall and $F_{0.5}$ scores, and reference-free metrics, represented by the Scribendi score. Precision, Recall, and $F_{0.5}$ scores are computed using a modified version of the ERRANT scorer (Bryant et al., 2017), with the $F_{0.5}$ score assigning twice the weight to precision compared to recall. Additionally, a human evaluation experiment is planned for a subset of submitted results following the shared task. The official evaluation was carried out on the CodaLab competition platform[5] based on the GLEU score.

In Table 4, we reported the official GLEU and F0.5 scores of our system and those of the baseline approach. Scores outperforming the baseline are reported in green; scores lower than the baseline in red. We note that we outperform the baseline for most languages, but we obtained very poor results for German, Icelandic and Slovene.

The failure with German is easy to explain: during the prediction phase, the system's execution was interrupted, not predicting all the needed output. , preventing it from predicting all the necessary output. Unfortunately, this issue was not noticed by our team, and we submitted an incomplete file. For the publication of this paper, we reran our pipeline correctly and obtained a GLEU score of 75.49 for this language.

However, for Icelandic and Slovene, we encountered serious problems. At first glance, the output appeared deviant, with the same sentences being repeated over and over. Therefore, we decided to submit the Icelandic corpus as it was, without modification, and to use the XLM-RoBERTa-based system for the Slovene dataset.

5 Discussion

5.1 Heterogeneity of the MultiGEC datasets

It is worth noting that the corpora exhibit high variability in annotation, which is crucial to consider when utilizing the MultiGEC dataset. The variation across corpora helps explain why developing one model per corpus yields better results than using one model per language.

For example, the choice of whether or not to capitalize addresses can differ from one corpus to another. Additionally, the learners of the language who wrote the original essays may come from different backgrounds. For instance, among the four different corpora for Czech, there are essays written by native students from elementary and secondary schools (NatForm), informal website texts (NatWebInf), essays written by Romani ethnic minority children and teenagers (Romani), and essays written by non-native speakers (Romani). The errors produced by these different profiles of speakers are undoubtedly specific to their age and social context, and therefore, the corrections are as well.

5.2 LLaMA-3.0's Pathological Output for Icelandic and Slovene

Our hypothesis is that these languages are ill supported by the LLaMA 3.0 model. Although Meta claims that it has been pre-trained on over 30 languages with high-quality data, non-English data accounts for only about 5% of the total pre-training corpus.[6]

We noticed that there is some Slovene Wikipedia data in LLaMA 3.0 (Touvron et al., 2023), but we suspect that it may not be sufficient. Additionally, the MultiGEC training data for Slovene are very limited. We found no evidence that LLaMA 3.0 possesses knowledge of Icelandic. This is further supported by an analysis of the tokenization performed by LLaMA.

If we look at the average number of characters per token for each set of essays, English, which is the default language of LLaMA, has 3.90 characters/token, German has 3.04 characters/token, Italian has 2.77 characters/token, and Icelandic has 1.83 and 1.87 characters/token (IceEC and IceL2EC, respectively), which is the lowest of all the languages present in the shared task data.

The average word length is 4.48 characters in English, and 4.31 and 4.65 characters (IceEC and IceL2EC, respectively) in Icelandic. Therefore, the token length difference cannot be explained by a word length difference alone. We suspect that this may be part of the explanation for the patho-

[5] https://codalab.lisn.upsaclay.fr/competitions/20500

[6] https://ai.meta.com/blog/meta-llama-3/

Language	Source	GLEU Lattice	GLEU Baseline	F0.5 Lattice	F0.5 Baseline
Czech	NatWebInf	**65.06**	53.91	**56.24**	32.93
	Romani	**53.70**	48.35	**45.99**	37.65
	SecLearn	**49.95**	45.77	**48.61**	46.18
	NatForm	71.45	76.08	31.99	37.46
English	Write & Improve	**77.90**	75.15	**53.79**	41.58
Estonian	EIC	**44.02**	36.47	22.73	24.38
	EKIL2	**56.96**	51.12	**38.07**	31.13
German	Merlin	0.05	69.56	14.09	52.58
Greek	GLCIIC	**51.49**	45.07	**44.07**	43.07
Icelandic	IceEC	**83.92**	80.52	0.00	8.19
	IceL2EC	39.79	39.93	0.00	8.14
Italian	Merlin	**69.96**	65.13	40.59	42.64
Latvian	LaVA	**67.25**	48.86	**57.77**	44.69
Russian	RULEC-GEC	77.77	79.02	**38.16**	34.71
Slovene	Solar-Eval	54.34	58.96	5.52	29.45
Swedish	SweLL_gold	**59.88**	58.40	40.01	41.46
Ukrainian	UA-GEC	**74.00**	68.03	**51.29**	22.66

Table 4: Comparison of our results with the baseline model on the minimal edits task. Results outperforming the baseline are highlighted in bold green, while those underperforming the baseline are in red.

logical behavior of our models with respect to this language.

Another reason for the poor performances of LLaMA seems to be the length of the essays. For all corpora except the Icelandic and Slovene ones, the average essay length is below 500 tokens (calculated with LLaMA's tokenizer), ranging from 33.6 tokens per essay on average for the Czech NatWebInf corpus to 460.8 tokens per essay for the Ukrainian UA_GEC corpus. English WriteAndImprove2024 essays is 188.8 tokens long on average. Slovene Solar-Eval essays are on average 1248.0 tokens long, the Icelandic IceL2EC essays are 1849.2 tokens long and the Icelandic IceEC essays are 2496.8 tokens long on average (calculated by the LLaMA tokenizer). Here again, the difference between Icelandic and English cannot simply be explained by the average token length difference. Icelandic essays are really longer than English ones and it seems that LLaMA has a harder time with longer inputs.

6 Distribution of Code

The code is available on GitHub under the MIT licence at the following address: https://github.com/lattice-8094/MultiGEC. It can be used to reproduce our results. Given the sensitivity of the data and the possibility of models leaking training data or hackers recovering the training data by inference attacks (Truex et al., 2021; Zhang et al., 2024), we will only distribute the program code. The data must be acquired by contacting the MultiGEC-2025 organizers. Researchers can obtain our models after making a personal request via email.

7 Conclusion and Future Work

We found that fine-tuning a multilingual large language model was a successful approach for most languages in the MultiGEC dataset, outperforming the baseline (using an LLM in a zero-shot setting). However, the model to be fine-tuned should have a minimal amount of knowledge about each target language for success. In this regard, we encountered difficulties with Slovenian and Icelandic.

Recently, we came across the Goldfish models (Chang et al., 2024): monolingual language models for 350 languages, including Icelandic and Slovenian, which propose smaller language models but with higher-quality data. It would be interesting to repeat the experiment using the Goldfish models and investigate whether the results for under-resourced languages can be improved.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jen-

nings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. 2024. Lora learns less and forgets less.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Christopher Bryant, Zheng Yuan, Muhammad Reza Qorib, Hannan Cao, Hwee Tou Ng, and Ted Briscoe. 2023. Grammatical error correction: A survey of the state of the art. *Computational Linguistics*, page 1–59.

Tyler A Chang, Catherine Arnett, Zhuowen Tu, and Benjamin K Bergen. 2024. Goldfish: Monolingual language models for 350 languages. *arXiv preprint arXiv:2408.10441*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale.

Christopher Davis, Andrew Caines, Oistein Andersen, Shiva Taslimipoor, Helen Yannakoudakis, Zheng Yuan, Christopher Bryant, Marek Rei, and Paula Buttery. 2024. Prompting open-source and commercial language models for grammatical error correction of english learner text. *ArXiv*, abs/2401.07702.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Mengsay Loem, Masahiro Kaneko, Sho Takase, and Naoaki Okazaki. 2023. Exploring effectiveness of GPT-3 in grammatical error correction: A study on performance and controllability in prompt-based methods. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 205–219, Toronto, Canada. Association for Computational Linguistics.

Arianna Masciolini, Andrew Caines, Orphée De Clercq, Joni Kruijsbergen, Murathan Kurfalı, Ricardo Muñoz Sánchez, Elena Volodina, and Robert Östling. 2025. The MultiGEC-2025 shared task on multilingual grammatical error correction at NLP4CALL. In *Proceedings of the 14th Workshop on Natural Language Processing for Computer Assisted Language Learning*, Tallin, Estonia. University of Tartu.

Grigori Sidorov, Anubhav Gupta, Martin Tozer, Dolors Catala, Angels Catena, and Sandrine Fuentes. 2013. Rule-based system for automatic grammar correction using syntactic n-grams for English language learning (L2). In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 96–101, Sofia, Bulgaria. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. 2021. Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing*, 14(06):2073–2089.

Elena Volodina, Christopher Bryant, Andrew Caines, Orphée De Clercq, Jennifer-Carmen Frey, Elizaveta Ershova, Alexandr Rosen, and Olga Vinogradova. 2023. MultiGED-2023 shared task at NLP4CALL: Multilingual grammatical error detection. In *Proceedings of the 12th Workshop on NLP for Computer Assisted Language Learning*, pages 1–16, Tórshavn, Faroe Islands. LiU Electronic Press.

Zheng Yuan and Mariano Felice. 2013. Constrained grammatical error correction using statistical machine translation. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 52–61, Sofia, Bulgaria. Association for Computational Linguistics.

Xinhao Zhang, Olga Seminck, and Pascal Amsili. 2024. Remember to forget: A study on verbatim memorization of literature in large language models. In *Proceedings of the Fifth Conference on Computational Humanities Research*, Aarhus, Danmark.