Grammatical Error Correction via Sequence Tagging for Russian

Anonymous ACL submission

Abstract

We introduce a modified sequence tagging architecture, proposed in (Omelianchuk et al., 2020), for the Grammatical Error Correction of the Russian language. We propose language-specific operation set and preprocessing algorithm as well as a classification scheme which makes distinct predictions for insertions and other operations. The best versions of our models outperform previous approaches and set new SOTA on the two Russian GEC benchmarks – RU-Lang8 and GERA, while achieve competitive performance on RULEC-GEC.

1 Introduction

Grammatical Error Correction (GEC) is the task of converting a source text to its correct variant so that it does not contain any grammatical, punctuation, spelling and lexical errors. Several types of models have been suggested as solutions for this task. Earlier studies concentrated on the most common error types in non-native English texts, e.g. incorrect choice of prepositions or determiners, and built error-specific classifiers (Chodorow et al., 2007; De Felice and Pulman, 2008). The development of deep learning and the invention of Transformer (Vaswani et al., 2017) led to a paradigm shift, and researchers began treating grammatical error correction, being a text-to-text task, as translation from the "language with errors" to the "grammatically correct language". Consequently, standard models for machine translation (MT), such as Transformer, were used for GEC task without adaptation. These models were trained on large corpora of parallel data, containing pairs of source sentences and their corrected versions (Grundkiewicz et al., 2019; Náplava and Straka, 2019).

Despite being fruitful and successful, especially during the BEA-2019 Shared Task for the English language (Bryant et al., 2019), this approach does not take into account the crucial difference between GEC and machine translation: in case of MT, source and target texts are not superficially related. These texts may even use different alphabets. However, the correspondence between initial texts and target texts in GEC is less arbitrary. Most of the words remain the same during the correction and the ones subject to modification often do not change their positions.

Moreover, single word edits are also restricted. For example, in case of morphological errors the correct word form belongs to the same lexeme and may be selected from the finite list of the source word inflections. Given all of this, the ability of sequence-to-sequence models to generate arbitrary texts is redundant during the GEC task and may even be detrimental due to the changes in the meaning of the text. Besides, machine translation models require large quantities of training data and are completely uninterpretable without external tools, which makes it complicated to apply them for educational purposes (Bryant et al., 2023).

Due to these considerations, it might be beneficial to formalize GEC as a sequence labeling task as opposed to the sequence transduction task. Instead of generating the target text, the sequence labeling model predicts individual word edits that transform the original sequence of words into the correct one. This approach was proposed in the seminal GECToR paper (Omelianchuk et al., 2020) for the English language, achieving the state-of-the-art performance at the time of publication (2020). In addition to its high quality, the GECToR approach has other benefits: sequence labeling is much faster than sequence transduction and requires less data to converge during the training. It is also more interpretable than the conventional sequence generation as individual edit operations correspond to common error patterns, such as choosing a wrong word form or an incorrect preposition.

Unfortunately, this interpretability does not come for free: the more complex is the morphology of the language, the more labour is required to design the label system reflecting it. Because of this, we know few equivalents of GECTOR for other languages than English: Chinese (Zhang et al., 2022), Ukrainian (Bondarenko et al., 2023), Arabic (Kwon et al., 2023) and Turkish (Kara et al., 2023).

We fill this gap by creating a GECToR-like model for Russian and demonstrate state-of-the-art performance on the two Russian GEC benchmarks out of three. We make our code available[1]. Our main contributions are as follows:

- We develop the label inventory and preprocession that take into account the complexity of Russian morphology.
- We present a modified classification schema which makes a distinction between insertions and other types of corrections. Moreover, we adopt a Large Language model for spelling correction.
- We conduct several experiments varying encoders, the size of synthetic data during the pretraining stage and the presence of token type embeddings, and achieve state-of-the-art results on the two Russian benchmarks: RU-Lang8 (Trinh and Rozovskaya, 2021) and GERA (Sorokin and Nasyrova, 2025), as well as competitive performance on the remaining one – RULEC-GEC (Rozovskaya and Roth, 2019).

2 Related Work

One of the first approaches to GEC was to design error-specific classifiers, for example, for the choice of prepositions, articles, verb or noun forms (Han et al., 2006; Chodorow et al., 2007; De Felice and Pulman, 2008; Tajiri et al., 2012; Rozovskaya et al., 2014; Berend et al., 2013; van den Bosch and Berck, 2013). These error types implied finite confusion sets, so it was relatively convenient to model them as classification among the corrections known in advance (Bryant et al., 2023). However, the classifiers for narrow domains were not able to correct other error types. They also could not be built for cases that did not have limited lists of corrections, for example, lexical choice errors, and relied excessively on the local context (Bryant et al., 2023).

Some of these limitations have been overcome by MT models which generated corrected texts based on their incorrect versions. Machine Translation GEC models were able to correct several error types simultaneously as well as interacting errors[2]. Initially, statistical machine translation models were implemented (Felice et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2014). The introduction of Transformer (Vaswani et al., 2017) has become an impetus for the development of neural machine translation (NMT), resulting in the success of NMT approach (Grundkiewicz et al., 2019) during the BEA-2019 Shared Task on Grammatical Error Correction (Bryant et al., 2019). However, the main shortcoming of MT models remained even in neural approaches – their dependency on the size and quality of training data. In (Náplava and Straka, 2019) machine translation models were considered for low-resource GEC: in Czech, German and Russian. The authors achieved higher performance in the two former settings because of the larger quantity of annotated data for these languages, than for Russian, despite pretraining on the same size of synthetic data for all three languages, which proves the crucial role of the size of data for MT approaches. Besides, MT models lack interpretability, it is difficult to comprehend why they do and do not correct certain errors and, consequently, use them in education (Bryant et al., 2023).

Sequence labeling architecture GECToR proposed in (Omelianchuk et al., 2020) is a much more efficient and interpretable solution than MT methods. According to GECToR, each token is assigned an operation label, so that after all operations are implemented, the correct version of a sentence is obtained. This approach highlights the global difference between GEC and MT, which is that most tokens in a sentence remain unchanged after the correction. Moreover, operation labels which correspond to common corrections, e.g. ‘convert the noun to its plural form’, are accessible and transparent. The operations consist of word-level edits, corresponding to insertion, deletion and replacement operations. In addition to these *basic transformations*, there are task-specific *g-transformations*. They include noun number and verb form changes.

Recent approaches to GEC also involve Large Language Models (LLMs). Their abilities were studied in zero-shot and few-shot settings (Wu et al., 2023; Fang et al., 2023; Loem et al., 2023) as well as after instruction-tuning on the grammatical

[1] The link will be added in the final version of the paper.

[2] For example, in some languages when a preposition is corrected, the case of the noun, which is governed by it, also has to be corrected.

error correction task (Kaneko and Okazaki, 2023; Omelianchuk et al., 2024). According to (Omelianchuk et al., 2024), LLMs and conventional methods appear complementary, so the best solution for English GEC now is to combine them in ensembles.

3 GECToR for Russian

3.1 Preprocession

Since grammatical error correction in GECToR (Omelianchuk et al., 2020) is formalized as a sequence labeling task, the initial step is to preprocess annotated data so that all tokens in a sentence – words or punctuation marks – are assigned an edit label. The standard format for GEC data is .M2, consisting of a tokenized source sentence and error annotations which contain offsets of erroneous sequences, error types and corrections (see ex.1)[3].

(1)

```
S He have driven car yesterday .
A 1 3|||Verb:form|||drove
A 3 3|||Det|||a
```

As errors and corrections in annotations may consist of multiple words, we cannot achieve a one-to-one correspondence between erroneous tokens and corrections based on just the annotation. Moreover, different corpora adopt distinct error type labels, so they cannot be used as operation labels and a universal preprocession algorithm is required. We refer to the Figure 1 for the description of label extraction. To implement it, we develop an algorithm of linguistic alignment, which is a modification of Levenshtein distance algorithm that has penalties for different lemmas and parts of speech and also accounts for merged-separate-hyphenated spelling of words. In order to obtain lemmas, parts of speech and morphological features, DeepPavlov/morpho_ru_syntagrus_bert[4] is used, being a high-quality morphosyntactic parser for Russian.

We follow (Omelianchuk et al., 2020) and construct a set of operation labels. However, for our model we create a modified label inventory to tackle the morphological complexity of Russian, as for a language with a large number of grammatical categories the number of g-transformations grows exponentially. Besides, in the English GECToR model a relatively large label set of 5000 operations is used, the majority of which represents replacements, corresponding to spelling errors. To reduce vocabulary size and make model training easier, we follow (Mesham et al., 2023) and predict a dedicated SPELL tag for spelling errors. Their corrections are generated in the postprocessing phase, see subsection 3.2.2 below.

3.2 Model

3.2.1 Classification

The original GECToR model cannot handle word modification and inserting another word after it in one step, that is why the authors adopt an iterative approach (Omelianchuk et al., 2020), with most corrections being done during the first two iterations. We will also study iterative editing in C.2. However, we also differentiate the prediction of insertions (in place of spaces) and other operations (pertaining to words) to manage several operations for one token.

Our scheme is illustrated in the Figure 2. More precisely, we modify the conventional token classification task so that labels would be predicted not only for subtokens[5], but also for spaces between them. Several decisions had to be made for it to be possible.

Firstly, determining how to represent tokens and spaces. It is not evident, at first glance, whether using the first or the last subtoken of tokens would be the optimal way to represent them in GEC, as various error types may occur both in the beginning and in the end of the word form, e.g. spelling errors are frequently made within the stem, whereas grammatical errors primarily affect inflections. For implementation considerations and by following (Omelianchuk et al., 2020), we decide to use the embeddings of first subtokens as the representations of tokens. As for the spaces between the tokens, we choose as their representation the average of the immediate preceding and following embeddings.

Secondly, finding a convenient way of implementing this approach. We adopted the following strategy: after the tokenization, two numeral masks are created. The process is reflected as step 2 in Figure 2: the light yellow mask (left-mask or LM) and light purple mask (right-mask or RM). They have the same length of $2n + 1$, where n is a number

[3] There are other fields in .M2, but they are omitted for illustrative purposes and are not pertinent to the description.

[4] https://docs.deeppavlov.ai/en/0.17.0/features/models/morphotagger.html#

[5] We use *subtokens* for units after the tokenization, as they may represent parts of tokens – symbols, word forms or punctuation marks.

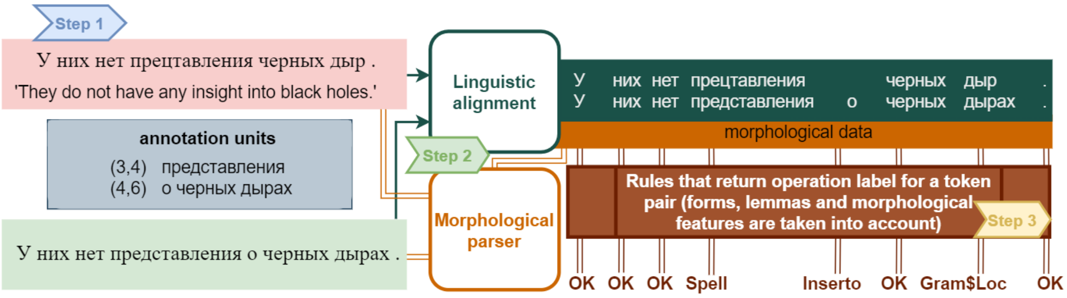
Step 1
У них нет прецтавления черных дыр .
'They do not have any insight into black holes.'
annotation units
(3,4) представления
(4,6) о черных дырах
У них нет представления о черных дырах .
Linguistic alignment
Step 2
Morphological parser
У них нет прецтавления черных дыр .
У них нет представления о черных дырах .
morphological data
Rules that return operation label for a token pair (forms, lemmas and morphological features are taken into account)
Step 3
OK OK OK Spell Inserto OK Gram$Loc OK

Figure 1: Our preprocessing pipeline. 1. Collecting a grammatical variant of source sentence, using error indices and corrections from annotation units. Source sentence is highlighted with light red, while target sentence – with light green. 2. Both sentences are passed through the morphological parser and linguistic alignment algorithm. As a result, pairs of corresponding tokens are gathered (word columns highlighted with emerald) as well as their morphological features and lemmas. 3. Adopting the information collected during the step 2, rules assign each token in the source text an operation label, so that if all operations are implemented, the source text would be transformed into the target sentence. E.g. in the given sentence only three non-KEEP operations are required: correcting a spelling error in *prectavleniya* 'insight', inserting *o* 'into' after it and changing the case of noun *dyr* 'holes' to locative. N.B. KEEP is replaced with OK in the figure for illustrative purposes.

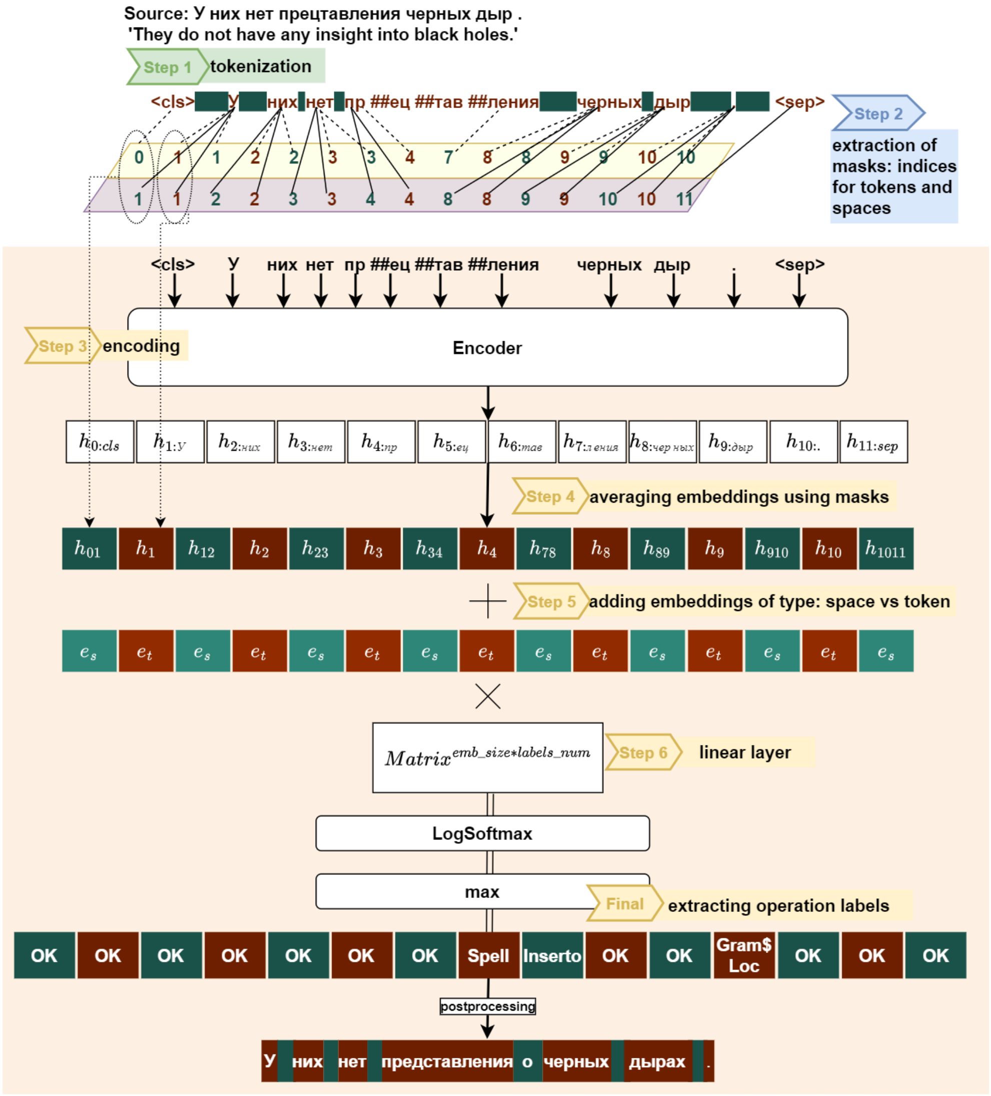
Source: У них нет прецтавления черных дыр .
'They do not have any insight into black holes.'
Step 1 tokenization
<cls> У них нет пр ##ец ##тав ##ления черных дыр . <sep>
0 1 1 2 2 3 3 4 7 8 8 9 9 10 10
1 1 2 2 3 3 4 4 8 8 9 9 10 10 11
Step 2 extraction of masks: indices for tokens and spaces
<cls> У них нет пр ##ец ##тав ##ления черных дыр . <sep>
Step 3 encoding
Encoder
$h_{0:cls}$ $h_{1:У}$ $h_{2:них}$ $h_{3:нет}$ $h_{4:пр}$ $h_{5:ец}$ $h_{6:тав}$ $h_{7:ления}$ $h_{8:черных}$ $h_{9:дыр}$ $h_{10:.}$ $h_{11:sep}$
Step 4 averaging embeddings using masks
h_{01} h_1 h_{12} h_2 h_{23} h_3 h_{34} h_4 h_{78} h_8 h_{89} h_9 h_{910} h_{10} h_{1011}
Step 5 adding embeddings of type: space vs token
e_s e_t e_s e_t e_s e_t e_s e_t e_s e_t e_s e_t e_s e_t e_s
$Matrix^{emb\_size*labels\_num}$
Step 6 linear layer
LogSoftmax
max
Final extracting operation labels
OK OK OK OK OK OK OK Spell Inserto OK OK Gram$ Loc OK OK OK
postprocessing
У них нет представления о черных дырах .

Figure 2: Our model pipeline.

of tokens in a source sentence. It accounts for all tokens, spaces after them and a space in the beginning as an insertion may be there as well. Numbers in dark green font represent spaces, whereas others (in dark brown font) – tokens. LM contains indices of first subtokens of tokens and of spaces' immediate preceding subtokens. RM consists of the former and of spaces' immediate following subtokens. For each of the $2n + 1$ spaces and tokens, a pair of left index and right index would become available: for tokens they would be expressed by the same number, whereas for spaces – by the indices of surrounding left and right subtokens. Afterwards, when a tokenized sentence is passed through an encoder and subtoken embeddings are obtained (step 3), masks are used to select only the embeddings of corresponding subtokens, consequently, there are two sets of embeddings: for subtokens 1) from LM and 2) from RM, which are then being averaged (step 4). As a result, $2n + 1$ embeddings are extracted, every second one corresponds to the token in a source text, others – to the spaces for insertions. Token embeddings are first subtoken embeddings, while space embeddings are the averages of surrounding subtokens' embeddings.

Thirdly, our preliminary research showed that models tend to confuse labels for spaces with labels for tokens, that is why we decide to add trainable embeddings of token type, representing spaces or tokens, and combine them (step 5) with subtoken embeddings from the previous step, effectively solving the issue.

3.2.2 Edit postprocessing

After predicting the labels, the corresponding output words are inferred. Most transformations are implemented with the help of rules. For grammatical labels we utilize the pymorphy2 library (Korobov, 2015) and its *inflect* method that allows to predict any inflected form of a word given the morphological features of the inflected word. In order to apply this function, we manually convert CoNLL-U morphological labels predicted by the DeepPavlov parser to the Pymorphy format.

For spelling labels we use the external API, namely YandexGPT[6]. We replace the words, preliminarily labeled with SPELL by the SPELL token and pass both source and the tagged sentence using the prompt given in the Figure 3. We decide to use a large language model instead of local spellcheckers since one needs to select among several possible corrections and traditional models do not provide such possibility. The LLM's response is verified and edited so that it complies with the following conditions: 1) The number of corrections corresponds to the number of submitted words with typos. 2) Corrections are close in Levenshtein distance and length to the source words. Otherwise, the source word remains unchanged. 3) Corrections do not contain unnecessary characters, such as arrows or brackets. 4) There are no markdown[7] elements, for example, ** to highlight in bold.

4 Model Evaluation

4.1 Data

Five existing Russian GEC datasets were used in the experiments: RULEC-GEC(Rozovskaya and Roth, 2019), RU-Lang8(Trinh and Rozovskaya, 2021), GERA(Sorokin and Nasyrova, 2025), RLC-GEC and RLC-Crowd((Kosakin et al., 2024)).

- RULEC-GEC is a subset of the RULEC Corpus(Alsufieva et al., 2012) that contains essays of 12 learners of Russian as a foreign language and 5 heritage speakers.
- RU-Lang8 is the Russian learner subset of Lang-8 Corpus(Mizumoto et al., 2012), which includes small texts produced by speakers of more than 34 languages. Only validation and test samples of RU-Lang8 were manually reannotated, while training data remains noisy, so the usage of this corpus in our experiments is reduced to these partitions.
- GERA is based on Russian middle school essays, representing the only source of Russian native speakers' errors.
- RLC-GEC and RLC-Crowd are derived from the Russian Learner Corpus (RLC)(Rakhilina et al., 2016), consisting of texts written by college and university learners of the Russian language from different countries. The former dataset is the subset of RLC which contains annotated corrections, whereas the latter consists of crowdsourced annotations.

Datasets vary greatly in error distribution and size, see Table 1. While spelling errors are the most prominent in RULEC-GEC and RU-Lang8, in GERA corrections of punctuation form the largest share. The RLC dataset is the only one that has

[6] https://yandex.cloud/ru/docs/foundation-models/concepts/yandexgpt/models

[7] http://daringfireball.net/projects/markdown/

> “Дорогая модель, тебе будут даны слова с опечатками, в скобках будет указано предложение, в котором они встретились. Пожалуйста, выведи исправления этих слов в том же порядке, но без предложения в скобках и каких-либо комментариев, начиная со слова "Ответ:".”
> ‘Dear model, you will be given words with spelling errors, the sentence where they were encountered will appear in the brackets. Please, print the corrections for these words in the same order, but with no sentence in the brackets and any comments, starting with the word "Answer:".’

Figure 3: The prompt for spelling correction.

lexical choice errors as most common, and, unlike others, has a much larger fraction of syntactic errors than other corpora. We report the distribution of top-7 operation labels (after the preprocession from 3.1) in training collections in Appendix A.

We test our models on the test partitions of RULEC-GEC, RU-Lang8 and GERA.

4.2 Training

We train several models, varying the following conditions: the type of encoder, the addition of token type embeddings (TTE), and the size of synthetic data during the pretraining. We use either ruRoberta-large[8] or FRED-T5-1.7B[9] as an encoder-model (Zmitrovich et al., 2024). We choose these models because they are open-source and demonstrate great performance on benchmarks for the Russian language, such as Russian SuperGlue (Shavrina et al., 2020), which contains various tasks on general language understanding, RuCoLA (Mikhailov et al., 2022), a dataset of sentences with their binary acceptability judgements, as well as on the task of inappropriateness identification (Zmitrovich et al., 2024). Besides, training of these models is possible with our computational resources.

Following (Sorokin, 2022), we conduct training in two stages: firstly, we pretrain the models on a large amount of data (training samples of RULEC-GEC and GERA, validation partition of RU-Lang8, RLC-based datasets and synthetic data from (Sorokin, 2022)), then we finetune the model on the training sample (or validation in case of RU-Lang8) of the dataset in question and evaluate the model on its test partition. We investigate the effect of the number of synthetic sentences during the pretraining on performance: 20K, 100K, and 234K, since they have a more uniform error distribution than natural data, so it is not evident whether the largest number would be optimal.

Based on the training data, a dictionary of labels for classification is compiled. It contains operations that occur at least 5 times.

We report the optimal values of hyperparameters in the Appendix B.

4.3 Evaluation

4.3.1 Metrics

The models are evaluated using the M^2scorer script (Dahlmeier and Ng, 2012), which extracts the edits from the tokenized system outputs that have the maximum overlap with gold-standard annotations and calculates $F_{0.5}$-score which is a conventional evaluation metric for the GEC task since (Ng et al., 2014), where precision is considered more significant than recall because omitting a correction is not as harmful as proposing an erroneous correction.

4.3.2 Models

We compare our models with systems from previous works.

- Transformer (Náplava and Straka, 2019; Trinh and Rozovskaya, 2021): a fully trained MT encoder-decoder model.
- finetuned ruGPT-large [10] (Sorokin, 2022; Sorokin and Nasyrova, 2025)
- ruGPT+ranker (Sorokin, 2022; Sorokin and Nasyrova, 2025): an architecture consisting of a correction generation with a language model and a correction ranking model based on ruRoberta-large[11]
- rules+ranker (Sorokin, 2022; Sorokin and Nasyrova, 2025): A model similar to the previous one, but it uses rules for correction gen-

[8] https://huggingface.co/ai-forever/ruRoberta-large

[9] https://huggingface.co/ai-forever/FRED-T5-1.7B

[10] https://huggingface.co/ai-forever/rugpt3large_based_on_gpt2

[11] https://huggingface.co/ai-forever/ruRoberta-large

RULEC-GEC (learners)	RULEC-GEC (heritage)	RU-Lang8	GERA	RLC dataset
Spell (18.6) Noun:Case (14.0) Lex. (13.3) Lack (8.9)	Spell (42.4) Punct (22.9) Noun:Case (7.8) Lex. (5.5)	Spell (19.2) Noun:Case (12.6) Lex. (11.6) Punct (10.3)	Punct (42.5) Spell (23.6) Lex (13.6) Noun:Case (5.1)	Lex. (19.7) Spell (15.8) Syntax (13.8) Noun:Case (8.3)
12,480		4,412	6,681	31,519 (GEC), 34,150 (Crowd)

Table 1: Top-4 most common errors in Russian GEC datasets and numbers of sentences in each of the datasets. The data for the first three columns is obtained from (Trinh and Rozovskaya, 2021), statistics for GERA and the RLC dataset are adopted from (Sorokin and Nasyrova, 2025) and (Kosakin et al., 2024), respectively. "Lex." stands for lexical choice errors.

eration. This model and the previous one are state-of-the-art Russian GEC models.

In addition, we present as baselines the results of two instruction-tuned large language models:

- Qwen-2.5-7B-Instruct[12]: An open-source instruction-tuned model. It shows high-quality performance, especially among models of its size, on various leaderboards that evaluate the ability of models to solve a wide range of tasks, for example, on MERA[13] (Fenogenova et al., 2024).
- T-lite 1.0[14]: the Qwen-2.5-7B-Instruct model adapted to the Russian language with the help of additional training. This model demonstrates even higher quality on benchmarks for Russian in MERA than its predecessor.

Both LLMs were instruction-tuned for GEC on the same training collections as our models, using learning rate of 1e-5 and batch size of 32 during the pretraining and learning rate of 1e-6 while fine-tuning.

4.3.3 Results

The results of our experiments are presented in the Table 2. Firstly, state-of-the-art quality is achieved using the best version of GECToR for the case on two benchmarks out of three (RU-Lang8 and GERA), while on RULEC-GEC GECToR demonstrates comparable performance with LLMs and ruGPT+rerank pipeline. The most reliable corrections, reflected in maximum precision for two datasets, are predicted by rules+rerank model.

According to the recall metric, large language models appear optimal for RULEC-GEC and RU-Lang8, which comes as no surprise as they modify the text more freely than GECToR, whose corrections are limited to operations included in the dictionary during the training. However, it should be noted that the recall of GECToR models on GERA is comparable to the one of language models, and even exceeds it with iterative application. Since punctuation errors prevail in GERA, we can assume that language models have no advantage over GECToR in their detection.

Continuing the analysis of the results, we observe an ambiguous effect of the increase in synthetic data quantity. For RULEC-GEC and RU-Lang8 100K synthetic sentences are optimal, while on GERA for some models additional data improves the quality even further.

As for the type of encoder, on RU-Lang8 ruRoberta-large is more successful than FRED-T5. This result is less clear on GERA: models without the addition of token type embeddings consistently show lower quality with the ruRoberta-large encoder than with FRED-T5, while TTE models based on ruRoberta-large, on the contrary, have an advantage over similar systems based on FRED-T5. On RULEC-GEC ruRoberta-large surpasses FRED-T5 in most cases. We suggest that representations from ruRoberta-large are more suitable for classification, because it is initially an encoder model, unlike the encoder-decoder FRED-T5, whose encoder blocks are extracted for classification.

As was mentioned above, we also varied the addition of TTE. On GERA their presence significantly improves the quality of the models. On other corpora, their impact is inconsistent: if the encoder is ruRoberta-large, it is almost always positive, whereas in case FRED-T5 – only in half of

[12] https://huggingface.co/Qwen/Qwen2.5-7B-Instruct

[13] https://mera.a-ai.ru/ru/leaderboard

[14] https://huggingface.co/t-tech/T-lite-it-1.0

	Synthetic Data	RULEC-GEC			GERA			RU-Lang8		
Model	(only for GECToR)	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Transformer	-	63.3	27.5	50.2[1]		NA		55.3	28.5	46.5[2]
ruGPT	-	65.7	27.4	51.3[3]	73.4	23.4	51.4[4]		NA	
ruGPT+rerank	-	**73.7**	27.3	**55.0**[3]	78.4	44.4	68.0[4]		NA	
rules+ranker	-	66.5	28.6	52.6[4]	**86.1**	42.9	71.6[4]	**70.5**	29.1	54.8[4]
Qwen 7B	-	60.2	32.6	51.5	74.3	48.2	67.1	60.2	36.7	53.4
T-lite	-	61.0	**35.2**	53.2	76.3	49.4	68.8	62.5	**40.4**	56.3
GECToR Adaptations										
ruRoberta	synth20K	*66.6*	*23.8*	*49.0*	69.1	30.0	54.8	*61.5*	26.4	48.6
ruRoberta$_{TTE}$	synth20K	64.8	23.1	47.6	*75.0*	*50.2*	*68.3*	61.2	*31.7*	*51.6*
FRED-T5	synth20K	64.7	18.6	43.2	70.4	34.4	58.2	58.2	24.9	45.9
FRED-T5$_{TTE}$	synth20K	60.6	14.7	37.3	68.6	42.4	61.1	50.7	23.5	41.2
ruRoberta	synth100K	60.7	21.6	44.6	71.0	34.9	58.8	60.3	26.6	48.1
ruRoberta$_{TTE}$	synth100K	*65.3*	26.4	*50.4*	*75.8*	49.8	*68.6*	*62.4*	*32.9*	*53.0*
FRED-T5	synth100K	64.4	21.0	45.5	73.5	35.5	60.5	60.7	23.7	46.3
FRED-T5$_{TTE}$	synth100K	56.6	*27.0*	46.4	72.9	*50.4*	66.9	56.5	32.7	49.3
ruRoberta	synth234K	61.1	*25.8*	48.0	69.0	34.9	57.7	*63.0*	29.0	51.0
ruRoberta$_{TTE}$	synth234K	*68.3*	22.6	*48.7*	*78.2*	49.1	*69.9*	62.9	*31.3*	*52.3*
FRED-T5	synth234K	65.4	21.5	46.4	73.4	33.4	59.2	58.7	27.5	47.8
FRED-T5$_{TTE}$	synth234K	57.9	24.3	45.4	73.6	*49.4*	67.0	57.6	28.5	47.8
Iterative implementation of the best GECToR version for each corpus										
Iteration #2		67.0	28.4	52.6	80.4	51.4	72.2	65.0	36.5	56.2
Iteration #3		*67.2*	*28.7*	*53.0*	*80.5*	***52.2***	***72.7***	*65.4*	*37.4*	***56.9***

Table 2: Main results. Best results are highlighted in bold, the highest metrics in different experimental setups are in italics, the best GECToR results for each corpus are underlined. Suffix $_{TTE}$ denotes addition of token type embeddings. Previous results are obtained from: [1]–(Náplava and Straka, 2019), [2]– (Trinh and Rozovskaya, 2021), [3]–(Sorokin, 2022), [4]–(Sorokin and Nasyrova, 2025).

the case. We assume that it depends on the fraction of insertion errors in the corpus.

Following (Omelianchuk et al., 2020), we apply the best versions of our model iteratively and find that after the second iteration the quality improves even further. However, after the third application the increase in quality is less prominent.

5 Conclusion

We adapt sequence tagging architecture from (Omelianchuk et al., 2020) to the Russian language. To do this, we create a language-specific preprocessing algorithm and operation inventory; in addition, we propose a modified architecture for classification, distinguishing the prediction of operations for tokens and insertion operations, we also introduce label decoding using a large language model.

We conduct several experiments, varying the encoder model, the amount of synthetic data in pretraining, and the presence of token type embeddings, and find that the optimal encoder is ruRoberta-large, size of synthetic data – 100K sentences, and adding TTE is useful for corpora with a large fraction of insertions. On the two out of three Russian GEC benchmarks, the best versions of our models, applied iteratively, surpass the results of previous approaches, SOTA models and LLMs, which confirms the effectiveness of the GECToR approach for the Russian language as well.

We conduct an error analysis and ablation study in C.

Limitations

Our research is limited to the Russian language and we do not evaluate the effect of added modifications on the English GECToR. Moreover, the quality of our models significantly depends on the quality of classification, which suffers from underrepresentation of certain operations (e.g. lexical replacements) in the training data, which may be handled by generating more diverse synthetic sentences in the future.

Acknowledgments

This document has been adapted by Steven Bethard, Ryan Cotterell and Rui Yan from the instructions for earlier ACL and NAACL proceedings, including those for ACL 2019 by Douwe Kiela and Ivan Vulić, NAACL 2019 by Stephanie Lukin and Alla Roskovskaya, ACL 2018 by Shay Cohen, Kevin Gimpel, and Wei Lu, NAACL 2018 by Margaret Mitchell and Stephanie Lukin, BibTEX suggestions for (NA)ACL 2017/2018 from Jason Eisner, ACL 2017 by Dan Gildea and Min-Yen Kan, NAACL 2017 by Margaret Mitchell, ACL 2012 by Maggie Li and Michael White, ACL 2010 by Jing-Shin Chang and Philipp Koehn, ACL 2008 by Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, ACL 2005 by Hwee Tou Ng and Kemal Oflazer, ACL 2002 by Eugene Charniak and Dekang Lin, and earlier ACL and EACL formats written by several people, including John Chen, Henry S. Thompson and Donald Walker. Additional elements were taken from the formatting instructions of the *International Joint Conference on Artificial Intelligence* and the *Conference on Computer Vision and Pattern Recognition*.

References

Anna A Alsufieva, Olesya V Kisselev, and Sandra G Freels. 2012. Results 2012: Using flagship data to develop a russian learner corpus of academic writing. *Russian Language Journal*, 62(1):6.

Gábor Berend, Veronika Vincze, Sina Zarrieß, and Richárd Farkas. 2013. Lfg-based features for noun number and article grammatical errors. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 62–67.

Maksym Bondarenko, Artem Yushko, Andrii Shportko, and Andrii Fedorych. 2023. Comparative study of models trained on synthetic data for ukrainian grammatical error correction. In *Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP)*, pages 103–113.

Christopher Bryant, Mariano Felice, Øistein E Andersen, and Ted Briscoe. 2019. The bea-2019 shared task on grammatical error correction. In *Proceedings of the fourteenth workshop on innovative use of NLP for building educational applications*, pages 52–75.

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*, 49(3):643–701.

Martin Chodorow, Joel Tetreault, and Na-Rae Han. 2007. Detection of grammatical errors involving prepositions. In *Proceedings of the fourth ACL-SIGSEM workshop on prepositions*, pages 25–30.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 conference of the north american chapter of the association for computational linguistics: human language technologies*, pages 568–572.

Rachele De Felice and Stephen G. Pulman. 2008. A classifier-based approach to preposition and determiner error correction in L2 English. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 169–176, Manchester, UK. Coling 2008 Organizing Committee.

Tao Fang, Shu Yang, Kaixin Lan, Derek F Wong, Jinpeng Hu, Lidia S Chao, and Yue Zhang. 2023. Is ChatGPT a Highly Fluent Grammatical Error Correction System? A Comprehensive Evaluation. *arXiv preprint arXiv:2304.01746*.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 15–24, Baltimore, Maryland. Association for Computational Linguistics.

Alena Fenogenova, Artem Chervyakov, Nikita Martynov, Anastasia Kozlova, Maria Tikhonova, Albina Akhmetgareeva, Anton Emelyanov, Denis Shevelev, Pavel Lebedev, Leonid Sinev, Ulyana Isaeva, Katerina Kolomeytseva, Daniil Moskovskiy, Elizaveta Goncharova, Nikita Savushkin, Polina Mikhailova, Anastasia Minaeva, Denis Dimitrov, Alexander Panchenko, and Sergey Markov. 2024. MERA: A comprehensive LLM evaluation in Russian. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9920–9948, Bangkok, Thailand. Association for Computational Linguistics.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263, Florence, Italy. Association for Computational Linguistics.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting errors in english article usage by non-native speakers. *Natural Language Engineering*, 12(2):115–129.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The AMU system in the CoNLL-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 25–33, Baltimore, Maryland. Association for Computational Linguistics.

Masahiro Kaneko and Naoaki Okazaki. 2023. Reducing sequence length by predicting edit spans with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10017–10029.

Atakan Kara, Farrin Marouf Sofian, Andrew Bond, and Gözde Gül Şahin. 2023. Gecturk: Grammatical error correction and detection dataset for turkish. *arXiv preprint arXiv:2309.11346*.

Mikhail Korobov. 2015. Morphological analyzer and generator for russian and ukrainian languages. In *Analysis of Images, Social Networks and Texts: 4th International Conference, AIST 2015, Yekaterinburg, Russia, April 9–11, 2015, Revised Selected Papers 4*, pages 320–332. Springer.

Daniil Kosakin, Sergei Obiedkov, Ivan Smirnov, Ekaterina Rakhilina, Anastasia Vyrenkova, and Ekaterina Zalivina. 2024. Russian learner corpus: Towards error-cause annotation for L2 Russian. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 14240–14258, Torino, Italia. ELRA and ICCL.

Sang Yun Kwon, Gagan Bhatia, El Moatez Billah Nagoudi, and Muhammad Abdul-Mageed. 2023. Beyond english: Evaluating llms for arabic grammatical error correction. *arXiv preprint arXiv:2312.08400*.

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. *arXiv preprint arXiv:2305.18156*.

Stuart Mesham, Christopher Bryant, Marek Rei, and Zheng Yuan. 2023. An extended sequence tagging vocabulary for grammatical error correction. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1608–1619, Dubrovnik, Croatia. Association for Computational Linguistics.

Vladislav Mikhailov, Tatiana Shamardina, Max Ryabinin, Alena Pestova, Ivan Smurov, and Ekaterina Artemova. 2022. RuCoLA: Russian corpus of linguistic acceptability. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5207–5227, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2012. The effect of learner corpus size in grammatical error correction of ESL writings. In *Proceedings of COLING 2012: Posters*, pages 863–872, Mumbai, India. The COLING 2012 Organizing Committee.

Jakub Náplava and Milan Straka. 2019. Grammatical error correction in low-resource scenarios. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 346–356, Hong Kong, China. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *Proceedings of the eighteenth conference on computational natural language learning: shared task*, pages 1–14.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. Gector – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170.

Kostiantyn Omelianchuk, Andrii Liubonko, Oleksandr Skurzhanskyi, Artem Chernodub, Oleksandr Korniienko, and Igor Samokhin. 2024. Pillars of grammatical error correction: Comprehensive inspection of contemporary approaches in the era of large language models. *arXiv preprint arXiv:2404.14914*.

Ekaterina Rakhilina, Anastasia Vyrenkova, Elmira Mustakimova, Alina Ladygina, and Ivan Smirnov. 2016. Building a learner corpus for Russian. In *Proceedings of the joint workshop on NLP for Computer Assisted Language Learning and NLP for Language Acquisition*, pages 66–75, Umeå, Sweden. LiU Electronic Press.

Alla Rozovskaya and Dan Roth. 2019. Grammar error correction in morphologically rich languages: The case of russian. *Transactions of the Association for Computational Linguistics*, 7:1–17.

Alla Rozovskaya, Dan Roth, and Vivek Srikumar. 2014. Correcting grammatical verb errors. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 358–367.

Tatiana Shavrina, Alena Fenogenova, Anton Emelyanov, Denis Shevelev, Ekaterina Artemova, Valentin Malykh, Vladislav Mikhailov, Maria Tikhonova, Andrey Chertok, and Andrey Evlampiev. 2020. Russiansuperglue: A russian language understanding evaluation benchmark. *arXiv preprint arXiv:2010.15925*.

Alexey Sorokin. 2022. Improved grammatical error correction by ranking elementary edits. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11416–11429.

Alexey Sorokin and Regina Nasyrova. 2025. Gera: A corpus of russian school texts annotated for grammatical error correction. In *Analysis of Images, Social Networks and Texts*, pages 148–163, Cham. Springer Nature Switzerland.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for esl learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 198–202.

Viet Anh Trinh and Alla Rozovskaya. 2021. New dataset and strong baselines for the grammatical error correction of russian. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4103–4111.

Antal van den Bosch and Peter Berck. 2013. Memory-based grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 102–108.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Haoran Wu, Wenxuan Wang, Yuxuan Wan, Wenxiang Jiao, and Michael Lyu. 2023. ChatGPT or Grammarly? Evaluating ChatGPT on Grammatical Error Correction Benchmark. *arXiv preprint arXiv:2303.13648*.

Yue Zhang, Zhenghua Li, Zuyi Bao, Jiacheng Li, Bo Zhang, Chen Li, Fei Huang, and Min Zhang. 2022. Mucgec: a multi-reference multi-source evaluation dataset for chinese grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3118–3130.

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 The distribution of top-7 most common operations in the pretraining data.

We present the description in the Figure 4.

B Optimal Hyperparameter Values for GECToR training

The values are given in the Table 4. Despite the general number of epochs in the Table, we save and evaluate the checkpoint with the optimal value of *sent_accuracy* on the validation data. *Sent_accuracy* denotes the percentage of sentences which were fully classified correctly.

GERA: ruRoberta$_{TTE}$+synth234K

Error Type	P	R	F$_{0.5}$
spelling	88.5	63.7	82.1
punctuation	79.0	65.0	75.7
lexical choice	37.0	8.2	21.7
noun:case	69.2	41.5	61.1

RU-Lang8: ruRoberta$_{TTE}$+synth100K

Error Type	P	R	F$_{0.5}$
spelling	60.0	53.3	58.6
punctuation	55.4	67.5	57.5
lexical choice	36.1	9.8	23.5
noun:case	71.2	51.9	66.2

RULEC-GEC: ruRoberta$_{TTE}$+synth100K

Error Type	P	R	F$_{0.5}$
spelling	70.9	54.7	67.0
punctuation	65.3	11.1	33.0
lexical choice	47.2	6.6	21.2
noun:case	66.1	55.5	63.7

Table 3: Quality of the best GECToR adaptations on the main error categories.

C Error Analysis and Ablation study

C.1 Error Analysis

We evaluate the best versions of GECToR for each corpus with the help of RLC-ERRANT[15] (Kosakin et al., 2024) tool on the main error types in the Table 3.

C.2 Iterations

We evaluate the best versions of GECToR after the first and the second iterations in the Table 5. The correction improves for the vast majority of error types after the second iteration, as this helps the model to recognize a greater number of violations in the text, as well as to refine the already predicted modifications, which makes corrections in the text more consistent and reliable.

C.3 Token Type Embeddings

We select two models with the most prominent contrast in results between the basic configuration and the setup with the addition of TTE to learn which types of errors they affect the most.

The first model is FRED-T5+synth20K on RULES-GEC: its quality decreases by 5.9 points with TTE. The second model is ruRoberta-large+synth20K on GERA: its quality, on the con-

[15] `https://github.com/Russian-Learner-Corpus/annotator`

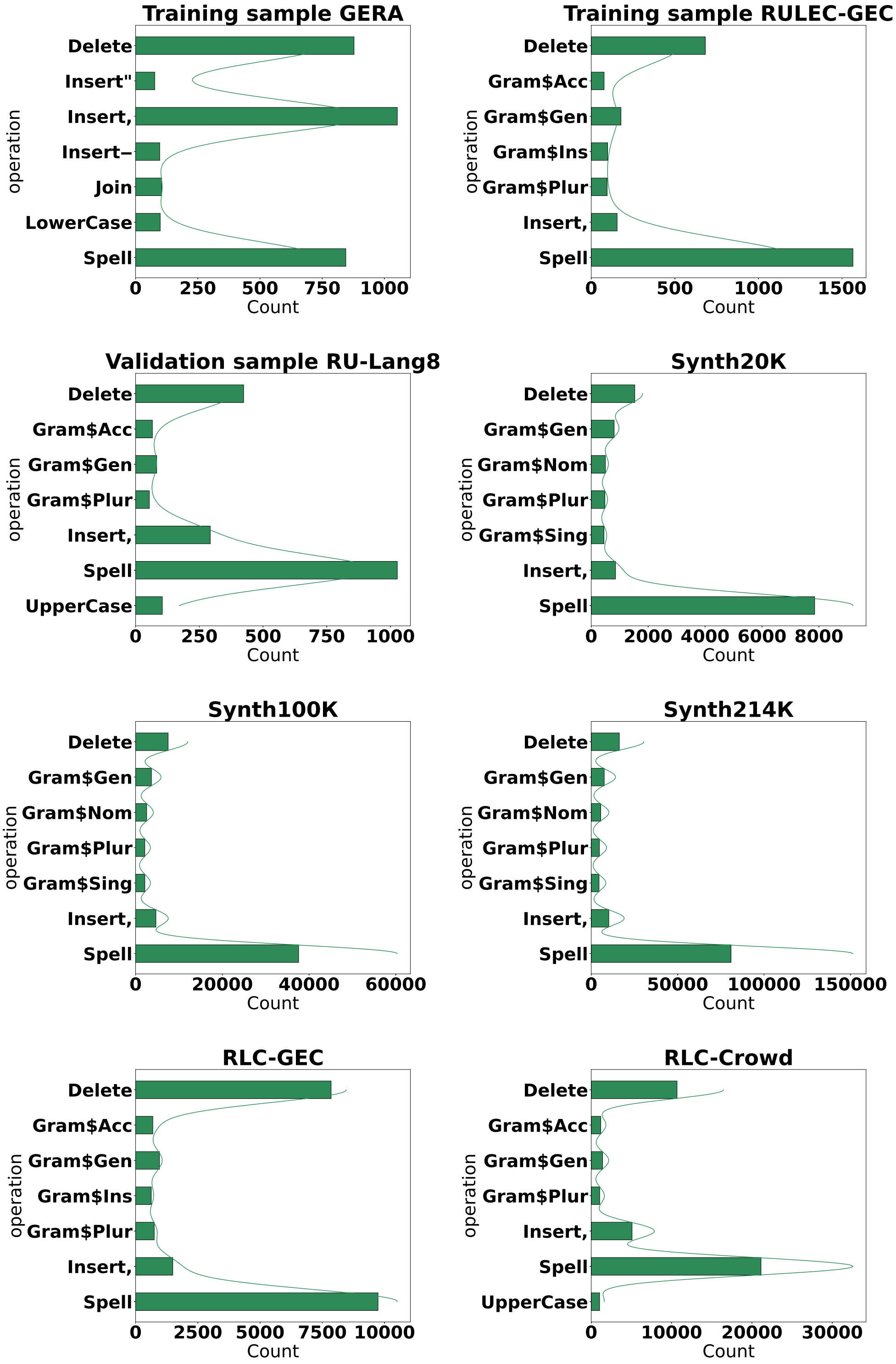

Figure 4: Top-7 most common operations in the samples which were used for training.

Hyperparameter	Encoder	
	ruRoberta-large	FRED-T5 1.7B
# epochs	3 (pretrain)/7 (finetune)	
batch_size	16	
learning rate	1e-05	1e-04
optimizer	AdamW	

Table 4: Optimal values of hyperparameters from our experiments.

GERA: ruRoberta$_{TTE}$+synth234K				**Iteration #2**		
Error Type	P	R	F$_{0.5}$	P	R	F$_{0.5}$
spelling	88.5	63.7	82.1	**89.7**	**67.4**	**84.1**
punctuation	79.0	65.0	75.7	**80.2**	**67.2**	**77.2**
lexical choice	37.0	8.2	21.7	**47.9**	**11.1**	**28.8**
noun:case	69.2	41.5	61.1	69.1	**44.6**	**62.2**

RU-Lang8: ruRoberta$_{TTE}$+synth100K				**Iteration #2**		
Error Type	P	R	F$_{0.5}$	P	R	F$_{0.5}$
spelling	60.0	53.3	58.6	**66.2**	**57.1**	**64.2**
punctuation	55.4	67.5	57.5	52.7	**69.3**	55.3
lexical choice	36.1	9.8	23.5	**39.3**	**12.9**	**27.8**
noun:case	71.2	51.9	66.2	70.5	**57.0**	**67.3**

RULEC-GEC: ruRoberta$_{TTE}$+synth100K				**Iteration #2**		
Error Type	P	R	F$_{0.5}$	P	R	F$_{0.5}$
spelling	70.9	54.7	67.0	**72.8**	**56.8**	**68.9**
punctuation	65.3	11.1	33.0	62.9	**12.7**	**35.1**
lexical choice	47.2	6.6	21.2	**47.3**	**7.6**	**23.1**
noun:case	66.1	55.5	63.7	**66.2**	**58.7**	**64.6**

Table 5: The comparison of the best models after the first and the second iterations. Improved results are highlighted in bold.

trary, increases by 13.5 points when they are added. A comparison of the models is shown in the Table 6.

D Classification

We calculate standard classification metrics for main operation types in the Table 7.

RULEC-GEC	**FRED-T5**			**FRED-T5$_{TTE}$**		
Error Type	P	R	F$_{0.5}$	P	R	F$_{0.5}$
spelling	73.9	**52.1**	**68.2**	**74.3**	50.3	67.8
punctuation	29.0	1.9	7.5	**56.9**	**13.5**	**34.7**
lexical choice	**48.3**	**6.1**	**20.3**	46.5	5.9	19.6
noun:case	**69.2**	**31.5**	**55.8**	45.7	16.0	33.3

GERA	**ruRoberta**			**ruRoberta$_{TTE}$**		
Error Type	P	R	F$_{0.5}$	P	R	F$_{0.5}$
spelling	**83.7**	61.1	**77.9**	80.5	**62.2**	76.1
punctuation	62.7	21.0	44.9	**75.6**	**68.4**	**74.0**
lexical choice	27.5	5.3	15.0	**34.8**	**7.7**	**20.5**
noun:case	63.6	43.1	58.1	**64.4**	**44.6**	**59.2**

Table 6: Comparison of models with and without TTE. The best results are highlighted in bold.

RULEC-GEC			
Operation Type	P	R	F_1
Delete	45.1	7.3	12.6
Gram	54.7	52.9	50.2
ReplaceFunc	62.2	39.2	44.3
ReplaceWord	0.0	0.0	0.0
ReplacePunct	0.0	0.0	0.0
Spell	69.3	42.1	51.7
Keep	97.9	99.7	98.8
Join	93.8	49.2	64.5
UpperCase	20.0	18.2	19.0
LowerCase	0.0	0.0	0.0
NullToHyphen	0.0	0.0	0.0
HyphenToNull	0.0	0.0	0.0
Insert,	82.6	12.9	22.3
Insertion	58.1	28.3	33.7

RU-Lang8			
Operation Type	P	R	F_1
Delete	54.5	19.6	28.8
Gram	58.7	58.5	57.1
ReplaceFunc	61.3	58.6	55.7
ReplaceWord	0.0	0.0	0.0
ReplacePunct	100.0	100.0	100.0
Spell	58.8	43.3	48.9
Keep	97.2	99.4	98.3
Join	56.2	47.4	51.4
UpperCase	35.4	68.0	46.6
LowerCase	78.9	57.7	66.7
NullToHyphen	0.0	0.0	0.0
HyphenToNull	0.0	0.0	0.0
Insert,	61.8	71.1	66.1
Insertion	63.6	35.4	36.0

GERA			
Operation Type	P	R	F_1
Delete	73.4	37.4	49.5
Gram	66.3	56.8	57.5
ReplaceFunc	100.0	33.3	50.0
ReplaceWord	0.0	0.0	0.0
ReplacePunct	33.3	25.0	28.6
Spell	75.9	43.5	54.9
Keep	98.6	99.8	99.2
Join	71.4	62.5	66.7
UpperCase	85.0	54.8	66.7
LowerCase	94.4	56.7	70.8
NullToHyphen	66.7	33.3	44.4
HyphenToNull	0.0	0.0	0.0
Insert,	85.7	82.2	83.9
Insertion	71.1	55.1	60.3

Table 7: Classification evaluation of the main operation types for the best GECToR models.