What Aggregate Scores Hide: Per-Rule Evaluation of Russian Grammatical Error Correction

Anna Smirnova[1]* **Artyom Kopan**[2] **Vladislav Makeev**[2] **George Chernishev**[2]
[1]HEC Lausanne, University of Lausanne [2]Saint Petersburg State University
anna.smirnova@unil.ch artyom.kopan@gmail.com
makeev.vladislav.d@gmail.com chernishev@gmail.com

Abstract

Russian grammar correction models can improve on aggregate benchmarks while getting worse at specific grammar rules. We show this through per-rule evaluation on a diagnostic benchmark of 48 prescriptive rules: finetuning on synthetic data improves overall $F_{0.5}$ while driving subordinate-clause comma accuracy from 14% to 1%. The suppression is invisible under corpus-level metrics and undetectable with existing coarse, corpus-specific tagsets; it is recoverable only when diagnosed at rule granularity. To enable this analysis, we develop a 98-category error taxonomy grounded in Rozental's reference grammar and SyntErr, an open-source synthetic data generator whose per-rule distribution is an explicit parameter, designed to support arbitrary rule sets and languages. Finetuning eight open models (0.8B–12B) on 39K synthetic examples yields up to 75.3 $F_{0.5}$, approaching frontier API models with models small enough to run on device. We release the taxonomy, generator, per-rule evaluation data, and all training artifacts.

1 Introduction

Russian grammatical error correction remains a small field with limited resources: the largest annotated corpus contains fewer than 15K sentences, and each of the handful of existing benchmarks defines its own error categories and tagset. These tagsets serve different purposes: RLC (Kosakin et al., 2024) classifies errors by their likely cause in the writer (e.g., case-government errors after a verb or preposition; transfer from the writer's first language, L1), which is useful for second-language (L2) pedagogy; RULEC-GEC and GERA use POS-based categories suited to automatic annotation. But neither approach identifies which prescriptive rule was violated.

Recent work has shown that generative LLMs are increasingly competitive with sequence labeling models for GEC, but they are still evaluated with the same aggregate metrics. This is problematic because the failure modes differ. Sequence-labeling models make one wrong prediction at a time, on individual tokens, which is easy to inspect. Generative models can systematically fail to apply a rule that was underrepresented in their training data, a failure that no aggregate score will reveal. Diagnosing the latter requires mapping errors to specific grammar rules.

We cross-reference three Russian GEC corpora against Rozental's reference grammar (Rozental', 1997), the most widely used prescriptive handbook for Russian writing norms[1], covering orthography, punctuation, and morphology/stylistics across 213 paragraphs. Coverage is sparse: the best corpus covers 65% of paragraphs, and 47 paragraphs have no attestation in any corpus.

A deeper problem is the data itself. RULEC-GEC and RU-Lang8, both L2 corpora, remain the primary training resources for Russian GEC, so current systems are optimized for learner errors. But L1 and L2 error distributions barely overlap outside of spelling: L2 corpora contain virtually no prescriptive rule violations, and some L2 errors superficially resemble L1 errors without sharing the same underlying cause. It has been shown (Sorokin and Nasyrova, 2025) that training on L2 data actively degrades performance on L1 rule-specific correction.

Our central argument is that GEC for morphologically rich languages suffers from three entangled problems that current approaches fail to decompose: (1) the sequence labeling formulation conflates rule identity, corpus frequency, and the specific replacement word into a single classification decision; (2) training on L2 corpora actively

* Corresponding author.

[1]For orthography and punctuation alone, the RAS-endorsed handbook by Lopatin et al. has greater institutional authority, but does not cover morphological norms, government, or stylistics, which are all essential for GEC.

Proceedings of the 21st Workshop on Innovative Use of NLP for Building Educational Applications, pages 463–478

suppresses L1 prescriptive rules because the two error populations are largely disjoint; (3) aggregate evaluation hides rule-level damage, making both problems invisible. We address all three through rule-grounded synthetic data (SyntErr), per-rule evaluation anchored in Rozental's reference grammar, and per-rule diagnostics that reveal training-data-induced suppression.

Our contributions are:

- **Taxonomy and cross-reference.** A three-level error taxonomy (8 categories / 29 subcategories / 98 fine-grained tags) grounded in Rozental's 213 paragraphs. We cross-reference three corpora (GERA, RULEC-GEC, LORuGEC) against this taxonomy and provide an empirical classification of 4,682 of GERA's 5,988 errors into our categories via LLM-assisted classification (Claude Sonnet 4.6), revealing per-corpus coverage gaps (§3).
- **SyntErr.** An open-source rule-based error generator that ties each synthetic error to a specific Rozental paragraph, with configurable per-rule error rates. The current release covers the 48 rules tested by LORuGEC (Sorokin and Nasyrova, 2025), a subset of our 98 categories that future work can extend.
- **Experimental evidence.** Fine-tuning eight open models (0.8B–12B) from three families on 39K SyntErr examples plus 348 LORuGEC examples reaches 75.3 $F_{0.5}$, approaching frontier APIs with on-device-sized models (§5).

2 Related Work

We review existing Russian GEC corpora (§2.1), synthetic data generation methods (§2.2), and position our approach relative to both.

2.1 Corpora

Table 1 summarizes the publicly available Russian GEC corpora. Tag-set granularity varies by more than an order of magnitude, from 4 coarse edit labels in RU-Lang8 to 57 cause-oriented tags in RLC-GEC. Most are L2 (learner) corpora; only GERA, LORuGEC, and CoRST contain L1 errors, and LORuGEC is the only one annotated at the level of individual grammar rules. Two of the larger resources (RLC, CoRST) are accessible only through a query interface rather than as bulk downloads. No corpus is annotated against a reference grammar.

RULEC-GEC (Rozovskaya and Roth, 2019), the main Russian GEC benchmark, contains 12.5K L2/heritage sentences annotated with 23 morphosyntactic tags. RU-Lang8 adds L2 author diversity but uses only four coarse edit labels. RLC-GEC (Kosakin et al., 2024) is a fully annotated subset of the Russian Learner Corpus with 57 cause-oriented tags. LORuGEC (Sorokin and Nasyrova, 2025) is the first Russian corpus annotated at the level of individual grammar rules (48 rules, 960 sentences).

Beyond error-annotated corpora, RuBLiMP (Taktasheva et al., 2024) provides 45K minimal pairs isolating 45 grammatical phenomena, showing that even large LMs fail on negation, reflexivity, tense, and aspect. RuCoLA (Mikhailov et al., 2022) offers 9.8K sentences for binary acceptability judgments. We use both as supplementary evaluation (§5).

2.2 Synthetic Data and Rule-Based Evaluation

Tagged corruption models (Stahlberg and Kumar, 2021, 2024) train a neural network to introduce errors guided by error type labels, but for Russian the only available labels come from 5K RULEC-GEC sentences with a coarse 20-tag set, so the generator reproduces L2 patterns and cannot generate L1 rule violations absent from its training data. Rule-based generation sidesteps this: GECTurk (Kara et al., 2023) (Turkish), Ma et al. (2022) (native Chinese GEC, closest to ours in motivation), Palma Gomez et al. (2023) (Ukrainian), and Náplava and Straka (2019); Náplava et al. (2022) (Czech) all define language-specific heuristics. Two recent studies anchor evaluation in reference grammars: Li et al. (2026) probe LLMs against 673 points from a Luxembourgish grammar book, finding that translation quality does not predict grammatical competence; Banno et al. (2026) use the English Grammar Profile to detect constructs in L2 writing. Sorokin and Nasyrova (2025) improve Russian GEC via retrieval-augmented prompting (41.0 → 56.1 $F_{0.5}$ on LORuGEC).

Across these resources and methods, three gaps recur. No Russian GEC corpus annotates errors against a reference grammar, so coverage of prescriptive rules is unknown. No Russian generator offers explicit per-rule control over the error distribution it produces; existing generators inherit their

Table 1: GEC corpora with natural Russian-language data

Corpus	Year	Size (annotated / full)	Source	Tag set size	Format
RLC-GEC	2024	31,519	L2	57	CSV
RLC	2013	∼90,000 / 193,189	L2/HL	46+	Query interface
RULEC-GEC	2019	12,480	L2/HL	23	M2
GERA	2024	6,681	L1, schoolchildren	31	M2
LORuGEC	2025	960	L1 grammar resources	48	M2
CoRST	2013	∼20,000 / ∼175,000	L1, students	39	Query interface
RU-Lang8	2021	4,412 / 51,575	L2	4	M2

distribution from the L2 corpora they were trained on. And evaluation reports single aggregate metrics that cannot separate gains in frequent rules from losses in rare ones. The rest of the paper takes up these three gaps in turn.

3 Analysis

3.1 Taxonomy

We anchor our error taxonomy in Rozental's Справочник по правописанию и стилистике (Rozental', 1997) (Handbook on Spelling and Style), the most widely used prescriptive handbook for Russian. It comprises three parts (orthography §§ 1–74, punctuation §§ 75–138, morphology/stylistics §§ 139–213) and consists of 213 paragraphs. We define a three-level taxonomy: 8 coarse categories (e.g., SPELL, PUNCT), 29 subcategories, and 98 fine-grained tags. Each fine-grained tag maps to one or more Rozental paragraphs and carries an applicability rating (full, partial, or L1-only). Parts I–II describe hard rules violated by both L1 and L2 writers. Part III is different: it covers morphological variants (e.g., § 198: в отпуске vs. the colloquial в отпуску), verbal government, agreement subtleties, and stylistic norms — prescriptive preferences that L2 learners rarely encounter. Of our 98 fine-grained tags, 55 are fully applicable to L2 writers, 28 partially, and 15 are L1-only. This makes the same taxonomy usable for evaluating both native-speaker and learner GEC systems.

3.2 Coverage Analysis

Existing corpora follow a corpus-driven error-set approach: tags are derived from errors already present in the data. The resulting tagsets are coarse. GERA's single PUNCT tag spans 16 distinct punctuation rules in our taxonomy; S:ORTH covers 24 spelling subcategories (Table 3). A model that corrects only doubled consonants but fails on root vowel alternations is indistinguishable under these tags. Prescriptive rules, by contrast, are explicit, reproducible, and citable: when a model fails on Rozental § 87 (comma in compound sentences), the failure traces to a specific norm. A full cross-reference (supplementary materials) confirms the scale of the mismatch: 77 of our 98 fine-grained tags fall under just 20 RULEC tags, and 95 fall under just 27 GERA tags.

To assess coverage at the level of individual rules, we built a per-paragraph evidence table[2] for three corpora: two L1 (GERA, LORuGEC) and one L2 (RULEC). The pipeline uses the Claude Sonnet 4.6 API: for each Rozental paragraph it searches the corpus for candidate error examples, classifies them to the specific sub-rule, and records the result. For GERA (5,988 errors) we additionally classified each error into our fine-grained tags, grouping errors by their original GERA tag so each classification request covered one tag at a time (Table 3). 4,682 errors received a fine-grained tag; the remaining 1,306 fell outside the 48 currently-implemented rules or the LLM declined to classify them. All assignments were reviewed by a single annotator (a native Russian speaker); approximately 12% of the model's initial assignments were corrected during review.

Table 2 shows that coverage is incomplete even at the paragraph level. Within each paragraph the situation is worse. A single Rozental paragraph may contain up to twenty sub-rules (the general rule, its exceptions, and special cases), but corpora typically attest only one of them, and sometimes only an exception rather than the general rule it modifies. The mapping is not clean in either direction: one correction can correspond to several rules, and one sentence can contain several distinct errors. RULEC has no punctuation coverage at all,

[2]Supplementary materials: https://synterr-nlp.github.io/papers/bea-2026/.

	GERA	RULEC	LOR	Any
I. Orth. (74 §§)	59	54	24	66
II. Punct. (64 §§)	41	0	11	42
III. Morph. (75 §§)	39	41	7	58
Total (213)	139	95	42	166

Table 2: Evidence coverage: Rozental paragraphs with at least one verified error example per corpus. RULEC has zero punctuation coverage. 47 paragraphs are unattested in any corpus. Condensed by Rozental chapter; the full per-paragraph table (213 rows) is in the supplementary materials.

GERA tag	N	N_{tags}	Top categories
PUNCT	2542	16	subordinate cl. (668), parenthetical (263), isolation (239)
S:ORTH	975	24	checked root (145), unchecked root (124), conjunction sp. (101)
G:NOUN:CASE	303	10	double gov. (110), prep. case (63)
S:LETTER:CASE	244	4	proper nouns (117), titles (63)
L:OTHER	233	3	paronyms (212)
L:REP	286	2	pleonasms (249)

Table 3: GERA error reclassification. Each GERA tag maps to multiple fine-grained categories in our 98-tag Rozental-grounded taxonomy. Automated classification by Claude Sonnet 4.6 with errors grouped by their original GERA tag so each request covered one tag at a time; results manually verified. N_{tags}: number of distinct fine-grained tags per GERA tag. GERA's single PUNCT tag spans 16 distinct punctuation rules; S:ORTH spans 24 spelling subcategories. Full distribution available at https://synterr-nlp.github.io/papers/bea-2026/.

and across all three corpora 47 paragraphs (about a fifth of Rozental) have no attestation.

The gaps are not random. Learner corpora (RULEC, RU-Lang8) systematically miss the L1-only rules, which is why training on L2 data has been shown to degrade L1 correction performance (Sorokin and Nasyrova, 2025). These gaps propagate directly into training data: a model trained on a corpus that lacks punctuation examples will not learn punctuation rules, regardless of architecture. With the shift toward LLM-based GEC this bottleneck becomes addressable. LLMs already command Russian grammar from pretraining; what they need is a rule-level training signal, not token rewrites, to improve on specific prescriptive norms.

3.3 Why Generative Models?

The sequence labeling formulation of GEC (classifying each token as KEEP, DELETE, REPLACE, or APPEND (Omelianchuk et al., 2020)) has been the dominant paradigm for six years: it guarantees minimal edits by construction, producing corrections that preserve the author's text rather than rewriting it. For English, where most corrections are single-token edits on a language with minimal inflection, the label vocabulary stays manageable.

For morphologically rich languages like Russian, the formulation entangles three things: the linguistic rule being violated, the frequency of that violation in the training corpus, and the specific replacement word. For example, the same case error gets a different label when it appears on книга than on стол: GECToR-style labels are concrete output forms (e.g., $REPLACE_книгу), not abstract operations like "put in the genitive," so each word × case combination demands its own label. GECToR-style models pick each token's correction from a fixed vocabulary of 5–20K discrete labels. This vocabulary grows combinatorially with case, number, gender, and animacy, and cannot generalize to rule–word combinations absent from training.

This class imbalance is severe: in both RULEC-GEC and GERA, 94% of source tokens are already correct and receive the KEEP label during training. A rule that appears in a handful of training examples can't compete with this signal: the model defaults to copying rather than correcting. This explains the finding of Sorokin and Nasyrova (2025) that fine-tuning on general GEC data is detrimental for rule-specific errors.

Generative models sacrifice the structural guarantee of minimal edits. In exchange, no single decision has to do all three jobs at once. The rule itself comes from pretraining. Which rules to train on comes from SyntErr, not from how often they appear in natural data. The editing behavior (*be precise, be minimal, do not rewrite*) comes from the prompt and training format, which makes it a soft constraint rather than an architectural one. Without careful prompt and data design, this softness leads to paraphrase; our overcorrection analysis (§6) quantifies this failure mode and how to mitigate it. In exchange, because the output is free text, any rule that SyntErr can generate becomes a training example with no change to the model.

4 SyntErr

We present SyntErr: an open-source error generation tool for GEC. SyntErr accepts any error taxonomy and any target distribution, extensible to new languages and rule sets. Given clean text, it introduces errors whose type, frequency, and direction are explicit YAML configuration parameters, not artifacts of corpus collection. Labels are correct by construction: each error is tagged with the rule that generated it.

Every step of the pipeline exploits linguistic information from stanza (Qi et al., 2020): POS tags and morphological features determine which tokens are valid corruption targets, dependency parses identify syntactic positions (clause boundaries for punctuation, `amod` arcs for agreement, `obl`/`nmod` for government targets), and pymorphy3 generates correctly inflected corrupt forms preserving all non-target features. The result is structurally realistic errors, not random noise. Concurrently, Qiu et al. (2025) reimplemented ERRANT using stanza for multilingual GEC annotation across five languages, validating stanza as the tool of choice for morphologically aware GEC pipelines.

Each example is a `(src, tgt, rule)` pair with one or several errors, depending on configuration. SyntErr's minimal-pair mode (one error per sentence) is what we use for supervised fine-tuning in this paper. The current release implements all 48 rules tested by LORuGEC (Sorokin and Nasyrova, 2025), generates 10K examples in under 2 minutes on CPU, and requires no GPU or annotated data as input.

Pipeline. The generation pipeline operates in four steps. (1) An error-type distribution is specified via YAML configuration, or set to uniform (default). (2) Clean source sentences are analyzed by stanza (Qi et al., 2020), which provides both morphological tags (contextual POS, case, gender, number, e.g., стали = noun.Gen or verb.Past) and a full dependency parse tree. The parse tree is critical: it tells the generator *which* tokens stand in a syntactic relation that a given rule can target. (3) An error type is selected according to the distribution; the dependency tree is queried for applicable positions (e.g., all `amod` arcs for adjective–noun agreement, all clause boundaries for comma rules), and one is chosen at random. (4) The corruption is applied using pymorphy3 for morphological inflection. The output is a `{src, tgt, rule}` triple. The pipeline diagram is shown in Figure 2.

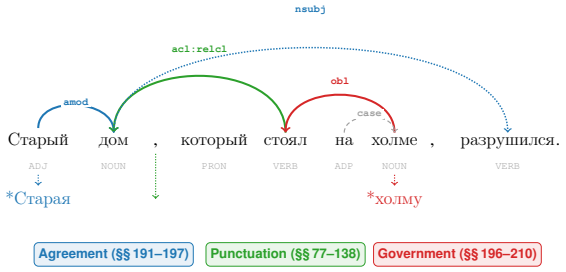

Figure 1: Dependency-driven error generation. Each arc family targets different syntactic relations: agreement (blue) uses `amod`/`nsubj`, punctuation (green) uses clause boundaries, government (red) uses `obl`/`nmod`. Corrupted forms shown below.

Most error types are not applied at random positions but at positions identified through dependency parsing: punctuation rules inspect clause boundaries and conjunction scopes, agreement rules traverse `amod` and `nsubj` arcs, and government rules use `obl`/`nmod` arcs to corrupt case frames (Figure 1). Spelling rules use dictionary lookup and Zaliznyak's stress dictionary (Zaliznyak, 2010); inflection uses pymorphy3. Detailed rule descriptions are in Appendix D.

5 Evaluation

5.1 Experimental Setup

Models. We evaluate models in three settings: (1) frontier API models: Claude Sonnet 4.6, Claude Opus 4.5/4.6, GPT-5.4, and GigaChat-2-Max (zero-shot only); (2) eight open models fine-tuned with LoRA: Qwen 3.5 (0.8B, 4B, 9B), Qwen 2.5-7B, Gemma 3 (1B, 4B, 12B), and Apertus-8B; (3) GigaChat-3.1-Lightning (10B MoE, 1.8B active; zero-shot only). All evaluations use greedy decoding.

Data. We generated 39,209 training examples across the 48 LORuGEC evaluation rules. Source text is drawn from the RuBLiMP source pool (Taktasheva et al., 2024) (38%), Taiga (Shavrina and Shapovalova, 2017) and news corpora (Fedorov, 2019) (25%), and targeted mining for forms required by underrepresented rules (e.g., rare conjunctions, diminutive suffixes, compound adjectives; 37%). The target distribution is approximately uniform; some rules are limited by source sentence availability. All RuBLiMP benchmark sentences are excluded. Punctuation corrections are skewed 3.6:1 toward comma insertion over deletion, reflecting the natural L1 tendency to omit

required commas rather than insert spurious ones.

Fine-tuning. We fine-tune with LoRA (Hu et al., 2022) (r=16, α=32, all attention and MLP projections), effective batch size 32, learning rate 10^{-4}, bf16 precision.

LORuGEC provides 348 validation and 612 test sentences. All reported scores are on the 612-sentence test set. For each model, we train three settings: (1) *SyntErr only* (39K synthetic examples), with early stopping on LORuGEC val loss (patience 5); (2) *LORuGEC only* (348 sentences — the LORuGEC val split, repurposed as training data), trained for 3 epochs; (3) *SyntErr then LORuGEC* (two-stage: first SyntErr with early stopping on val, then continued on the 348 examples for 3 epochs). Conditions (2) and (3) train on the LORuGEC val split; with 348 examples at effective batch size 32, training completes in ∼33 steps, too few to trigger intermediate evaluation. We additionally train a clean-text control where source equals target (no errors introduced), to verify that gains come from error patterns rather than task-format exposure.

Prompt. All training and evaluation use the system prompt proposed by Sorokin and Nasyrova (2025) for LORuGEC evaluation ("Дорогая языковая модель..."; full text in Appendix A). This prompt instructs the model to minimally correct the input text. We adopt it to ensure fair comparison with published baselines, but note that prompt sensitivity varies substantially across model families (Table 7). We additionally evaluate frontier models with a strict prompt that explicitly forbids word substitution ("НЕ МЕНЯЙ СЛОВА" — 'don't change words'; full text in Appendix A). The strict prompt is used only for the prompt-sensitivity ablation (Table 7), not for the main results in Table 4: it deliberately disallows lexical substitution, which is appropriate for spelling and punctuation rules but not for morphological or stylistic ones. The baseline prompt asks for minimal correction but does so in indirect, deferential language; some models interpret this as license to paraphrase. The strict prompt constrains the model to targeted edits only. The gap reaches 16.7 $F_{0.5}$ on GigaChat-2-Max (Table 7), confirming that prompt design is a major confound in GEC evaluation (Li et al., 2026).

Benchmarks. *LORuGEC test*: 612 test sentences, 48 prescriptive rules. *GERA*: 1,314 test sentences from school texts, OOD for synthetic data. *RuCoLA*: 1,804 OOD dev sentences for grammaticality judgment (Matthews Correlation Coefficient; 0 = chance, 1 = perfect). All GEC scores use M2scorer $F_{0.5}$.

We normalize ё→е and dash variants before scoring (details in Appendix E).

5.2 Results

Fine-tuning on SyntErr improves LORuGEC scores across all models (Table 4). Fine-tuning on SyntErr improves LORuGEC $F_{0.5}$ over zero-shot for every model tested. Qwen3.5-0.8B jumps from 13.7 to 45.2 (+31.5), Qwen3.5-4B from 47.6 to 67.0 (+19.4), and Apertus-8B from 48.7 to 69.4 (+20.7). Continuation on 348 real LORuGEC examples adds further gains: Qwen3.5-4B reaches 75.3 and Apertus-8B 72.4, approaching Claude Opus 4.6 zero-shot (81.8, Table 5) with far fewer parameters. For Qwen3.5-9B, SyntErr-only (57.3) still improves over zero-shot (49.2) but underperforms lorugec-only (61.2), likely due to the 3.6:1 comma insertion skew in training data; continuation training recovers to 70.9.

Clean-text control. To test whether SyntErr's gains come from learning error patterns or simply from exposure to the prompt-and-response format, we train on clean text where the source and target are identical, following the no-corruption baseline of Kiyono et al. (2019). This collapses performance: Qwen2.5-7B drops to 2.0 on LORuGEC (from 39.5 zero-shot), Qwen3.5-0.8B to 0.0, confirming that the gains require actual error examples.

Per-rule analysis. LORuGEC's 48 rules allow fine-grained evaluation. On Qwen3.5-4B, SyntErr→LORuGEC continuation improves 39 of 48 rules over zero-shot, degrades 5, and leaves 4 unchanged. Table 6 shows the extremes. Gains concentrate on fused/split spelling rules that SyntErr generates by construction; regressions concentrate on government and lexical rules that require semantic judgment. Notably, subordinate clause comma deletion drops from 10% to 0% under SyntErr-only: the 3.6:1 insertion skew in training data teaches the model to preserve commas rather than remove them. Continuation on 348 real examples recovers this to 50%. Full per-rule results (48 rules × 35 settings) are available at `https://synterr-nlp.github.io/papers/bea-2026/`.

Does GEC finetuning break general language understanding? Training a model to correct specific grammar rules could in principle make it reject

Model	Params	LORuGEC (612 sent.)				GERA (1,314 sent.)			
		ZS	+lor	+syn	+s→l	ZS	+lor	+syn	+s→l
Prior-work baseline[†]									
Qwen2.5-7B[†]	7B	39.5	44.0	58.1	**66.6**	47.6	44.5	45.5	**50.5**
Qwen 3.5 family									
Qwen3.5-0.8B	0.8B	13.7	33.8	45.2	**54.0**	25.8	38.0	**44.0**	36.9
Qwen3.5-4B	4B	47.6	54.7	67.0	**75.3**	56.5	47.4	47.9	**62.8**
Qwen3.5-9B	9B	49.2	61.2	57.3	**70.9**	56.8	54.4	52.4	**59.4**
Gemma family									
Gemma-3-1B	1B	18.9	25.5	**48.0**	47.8	26.6	27.6	**39.3**	32.2
Gemma-3-4B	4B	41.5	43.0	54.5	**58.2**	46.3	42.3	**48.9**	46.7
Gemma-3-12B	12B	52.5	58.8	66.0	**69.7**	53.8	55.4	**59.9**	57.9
Other									
Apertus-8B	8B	48.7	54.1	69.4	**72.4**	51.7	54.9	61.1	**62.1**
GigaChat-3.1[*††]	10B	52.0	—	—	—	60.4	—	—	—

Table 4: Finetuning results (M2 $F_{0.5}$). *ZS*: zero-shot (baseline prompt). *+lor*: LoRA on LORuGEC val (348 examples). *+syn*: LoRA on SyntErr (39K synthetic). *+s→l*: SyntErr then continued on LORuGEC val. [†]Same model used by Sorokin and Nasyrova (2025), who report 41.0 ZS / 56.1 with RAG (LORuGEC) and 67.1 finetuned (GERA). [*]MoE: 10B total, 1.8B active parameters. [††]ZS only; a 1K-token system preamble in the default chat template conflicts with the GEC instruction.

Model	LORuGEC	GERA
GigaChat-2-Max (baseline)	48.4	37.5
GigaChat-2-Max (strict)	65.1	58.7
GPT-5.4 (strict)	77.4	66.1
Claude Sonnet 4.6 (strict)	78.2	71.5
Claude Opus 4.5	79.8	64.1
Claude Opus 4.6	**81.8**	70.1

Table 5: Frontier API models, zero-shot (M2 $F_{0.5}$). Strict = explicit "do not change words" prompt.

Rule	ZS	+s→l	Δ
Top 5 improvements			
зато (spelling)	16.7	91.7	+75.0
Paired-conj. punctuation	8.3	83.3	+75.0
Dash in asyndetic sent.	6.7	80.0	+73.3
Particle -таки	7.7	76.9	+69.2
оттого (spelling)	33.3	94.4	+61.1
Regressions			
Prepositional case (управление)	50.0	37.5	−12.5
Suffixes -ек/-ик	55.6	44.4	−11.1
Parenthetical punctuation	57.9	47.4	−10.5
Suppression + recovery (SyntErr-only → continuation)			
Subord. clause commas (SyntErr-only: 0%)	10.0	50.0	+40.0

Table 6: Per-rule exact-match accuracy (%) on Qwen3.5-4B, LORuGEC test. ZS = zero-shot; +s→l = SyntErr then LORuGEC continuation.

sentences that native speakers find perfectly acceptable. To check this, we evaluate all models on RuCoLA (Mikhailov et al., 2022), a benchmark where the model judges whether Russian sentences are grammatically acceptable or not. For models ≥4B parameters, MCC remains stable across all training settings (Table 9). Smaller models (0.8B, 1B) show some degradation under SyntErr, reflecting the narrower capacity trade-off at that scale.

6 Discussion

GERA degradation. SyntErr's source text is formal, edited prose (news, encyclopedias); GERA contains school essays by children. Qwen models show GERA degradation after SyntErr training (Qwen3.5-4B: 56.5 → 47.9), but Gemma models do not (46.3 → 48.9). The pattern is consistent across model sizes within each family, suggesting an architecture- or pretraining-level difference rather than a capacity issue. On Qwen3.5-4B, SyntErr increases the share of unchanged outputs (*copy* in Table 8) on GERA from 7.7% to 22.5%: the model learns to be conservative from minimal-pair training and under-applies this to the noisier school-essay domain. Gemma, which starts with a lower zero-shot baseline, has less to lose from increased conservatism. Continuation on LORuGEC recovers GERA in all cases.

Prompt sensitivity. Replacing the baseline prompt with a strict variant yields up to +16.7 $F_{0.5}$ on GigaChat-2-Max (Table 7).

Reasoning hurts. Enabling thinking mode on Qwen3.5-9B drops LORuGEC $F_{0.5}$ from 49.2 to 23.4. The model cites Rozental by name in 165

	LORuGEC		GERA	
	Base	Strict	Base	Strict
GC-2-Max	48.4	65.1 +16.7	37.5	58.7 +21.2
GPT-5.4	73.7	77.4 +3.7	59.4	66.1 +6.7
Claude 4.6	76.7	78.2 +1.5	65.9	71.5 +5.6

Table 7: Prompt sensitivity (M2 $F_{0.5}$). Base = baseline prompt (Appendix A); Strict = explicit "do not change words."

	LORuGEC			GERA		
	Copy	Exact	Over	Copy	Exact	Over
ZS	31.2	28.3	40.5	7.7	40.5	51.8
+lor	5.1	23.5	71.4	3.1	13.0	83.9
+syn	38.4	38.4	23.2	22.5	55.0	22.5
+s→l	24.8	**54.4**	20.8	16.1	**60.3**	23.7

Table 8: Prediction behavior (%, Qwen3.5-4B). *Copy*: pred = source. *Exact*: pred = gold. *Over*: pred ≠ source ≠ gold. LORuGEC-only (+lor) produces 71–84% overcorrection.

of 612 examples, but states the rule incorrectly 91% of the time. On *и так, и эдак (no comma per § 154), it finds the correct answer (*"sometimes written without commas as a fixed idiom"*) but frames it as optional and copies the input. The minimize-changes instruction interacts adversarially with the reasoning process: the model uses it to justify rejecting every correction it considers (*"since the prompt says 'don't change correct punctuation,' I should not remove them"*). On rare words the model cannot recognize, reasoning produces empty output: up to 167 self-correction loops ("Wait, let me reconsider...") consuming 14K tokens without committing to an answer. These loops are a reasoning pattern trained via reinforcement learning on math and code (DeepSeek-AI, 2025), where backtracking enables error recovery; for GEC, they amplify uncertainty rather than resolving it. The model's prescriptive knowledge is too shallow to support productive chain-of-thought decomposition (Wei et al., 2022). The degradation is not category-specific either: under thinking, exact-match accuracy drops in every L0 category we measured (spelling −21.5, punctuation −22.4, agreement −25.6, morphology −20.0, government −56.2, lexical −10.7, averaged across the 35 LORuGEC rules mappable to our taxonomy). Mechanical categories where reasoning is plausibly unnecessary (spelling, punctuation) suffer at the same magnitude as semantic ones, ruling out the reading that thinking helps complex rules while hurting simple ones. Scaling does not help either: Qwen3.5-35B-A3B (MoE, 9B active) drops from 58.9 to 29.3 under the same conditions, confirming that reasoning-mode degradation is not a capacity issue.

The overcorrection problem. Sorokin and Nasyrova (2025) argue that generative models rewrite for fluency rather than minimally fixing errors. We quantify this directly (Table 8). On LORuGEC, zero-shot Qwen3.5-4B produces wrong edits 40.5% of the time (overcorrection) and copies the input unchanged 31.2% (undercorrection). SyntErr finetuning halves overcorrection to 23.2% and doubles exact match from 28.3% to 38.4%. The second training stage (continuation on real LORuGEC examples) teaches the model when *not* to edit: overcorrection drops to 20.8%, exact match reaches 54.4%. LORuGEC-only finetuning (348 examples) fails badly: 71.4% overcorrection, showing the model learns to edit everything but not *what* to edit.

Per-rule analysis reveals the mechanism: on rules SyntErr covers well, overcorrection is near zero (0–6%); on rules requiring lexical judgment, it remains 43–78% as the model normalizes register rather than correcting grammar.

Aggregate scores hide rule-level damage. Averaged across all eight models, SyntErr finetuning suppresses subordinate clause comma deletion from 14% to 1%: because SyntErr generates comma *insertion* errors 3.6× more often than *deletion* errors, the model learns to preserve commas rather than remove them. Meanwhile, compound-word spelling (e.g., оттого vs. от того) improves from 28% to 85%. Aggregate $F_{0.5}$ rises because the spelling gains outweigh the punctuation losses, but the punctuation rules are silently suppressed, invisible under aggregate evaluation. Continuation on 348 real examples recovers the affected rules (1% → 69%): the suppression is reversible once diagnosed. This is only possible with per-rule evaluation grounded in a reference grammar.

L1 transfer without L1 training data. Training exclusively on SyntErr (synthetic L1) and 348 LORuGEC examples transfers to GERA without any GERA-specific training: Qwen3.5-4B reaches 62.8 $F_{0.5}$, within 4.3 points of Sorokin and Nasyrova's Qwen2.5-7B finetuned *on GERA data* (67.1). SyntErr teaches which errors to fix; LORuGEC

continuation teaches when not to edit.

7 Conclusion

We have presented three contributions toward rule-grounded GEC for morphologically rich languages. First, a 98-category taxonomy anchored in Rozental's reference grammar, revealing that existing corpora cover at most 65% of its paragraphs and that their coarse tagsets mask critical gaps. Second, SyntErr, an error generator whose per-rule distribution is an explicit parameter, not an artifact of corpus collection; the current release covers 48 of the taxonomy's 98 categories and is open to extension. Third, experimental evidence across eight models from three families that 39K synthetic examples plus 348 real examples yield up to 75.3 $F_{0.5}$ on LORuGEC, approaching frontier models with 0.8B–12B open models.

Per-rule evaluation, enabled by the taxonomy, reveals what aggregate scores hide: training data can suppress specific rules (subordinate clause commas: 14% $\rightarrow$ 1%, averaged across models) while improving others (зато spelling: 17% $\rightarrow$ 92%), and continuation training on a small diagnostic set recovers the suppressed rules. This is the practical payoff of grounding GEC evaluation in a reference grammar.

A natural next direction is reinforcement learning rather than supervised fine-tuning. SFT remains the dominant recipe, but our results suggest it can distort the model's edit distribution: the model learns the marginal of the synthetic data ("edit something on every sentence") even where the input is already correct, which is exactly the overcorrection pattern we measure. A reward that captures minimal-edit behavior at the rule level should let the model retain pretrained linguistic knowledge while still picking up rule-specific corrections, without acquiring the SFT-induced bias toward editing.

We release the taxonomy with cross-reference tables, SyntErr, all training configurations and adapter weights, and per-rule evaluation data[3].

Limitations

Scope. We evaluate on Russian only. We do not evaluate on L2 (learner) benchmarks or at the document level, and we use a single prompt regime in the main results (§6 reports a separate prompt-sensitivity analysis but not a fully orthogonal sweep).

Coverage. Our taxonomy defines 98 fine-grained tags across 213 paragraphs; the current SyntErr implementation covers 48 (those tested by LORuGEC). The remaining 50 tags are implementable but not evaluated here. Punctuation generation is also directionally skewed (3.6:1 toward comma insertion; §5), which biases the kinds of punctuation errors the model ever sees during fine-tuning.

Single seed and training-stack sensitivity. We report single-seed numbers throughout and did not re-run training across multiple seeds. A small exploratory multi-seed run on two of the 4B configurations turned out to be inconclusive: the absolute scores depended substantially on the training stack (quantization, optimizer, kernel choices), and the cross-stack swing was larger than any plausible seed-to-seed variance, in a model-family-dependent way. LoRA fine-tuning at this scale is known to be sensitive to these choices, and disentangling stack effects from seed effects is beyond the scope of this paper. The absolute numbers in this paper should therefore be read with that envelope of uncertainty, even though the rule-level patterns we describe (suppression of specific rules under fine-tuning, recovery via continuation on a small diagnostic set) were consistent across all configurations we tried.

Stage order. Our two-stage models train on SyntErr first and then continue on LORuGEC. We did not test the reverse order or a randomly mixed schedule; stage order plausibly matters and is left to future work.

Validation as stopping signal. Our SyntErr-only models use the LORuGEC validation split as the early-stopping signal. This split is also used as training data in the +LORuGEC and SyntErr$\rightarrow$LORuGEC settings, so the val-as-stop and val-as-train roles are entangled; SyntErr-only numbers may be slightly optimistic relative to a fully held-out criterion.

Single-annotator categorization. The per-paragraph mapping between Rozental and the three corpora reflects one careful annotator's judgment; we do not report an inter-annotator agreement score. The cross-reference should be

[3]All artifacts are available at https://synterr-nlp.github.io/papers/bea-2026/.

read as a high-effort survey rather than a validated annotation.

Strict prompt is an ablation. The *strict* prompt (§6) restricts the model to orthography and punctuation and disallows lexical substitution by design. We use it to isolate prompt sensitivity and not as a competitive system; its scope is narrower than the full GEC task.

Ethics Statement

Our taxonomy encodes prescriptive norms as a reproducible evaluation tool, not as a claim about how Russian should be written. SyntErr uses no personally identifiable data; all evaluation corpora are publicly available.

Acknowledgments

We thank the anonymous reviewers for their feedback. We thank Simon Scheidegger (HEC Lausanne) for providing access to the DGX Spark used for training and evaluation. Claude (Anthropic) was used for English prose drafting, SyntErr and evaluation pipeline development, and LLM-assisted error classification. All generated text and code were manually verified by the authors.

References

Stefano Banno, Penny Karanasou, Kate Knill, and Mark Gales. 2026. Exploiting the English grammar profile for L2 grammatical analysis with LLMs. *arXiv preprint arXiv:2603.17171*.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Maxim Fedorov. 2019. Russian keyword evaluation datasets. https://github.com/mannefedov/ru_kw_eval_datasets.

Gemma Team, Aishwarya Kamath, Johan Ferret, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Hugging Face. 2023. PEFT: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

Atakan Kara, Farrin Marouf Sofian, Andrew Bond, and Gözde Şahin. 2023. GECTurk: Grammatical error correction and detection dataset for Turkish. In *Findings of the Association for Computational Linguistics: IJCNLP-AACL 2023 (Findings)*, pages 278–290, Nusa Dua, Bali. Association for Computational Linguistics.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. 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 1236–1242, Hong Kong, China. Association for Computational Linguistics.

Daniil Kosakin, Sergei Obiedkov, Ivan Smirnov, Ekaterina V 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.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Lujun Li, Yewei Song, Lama Sleem, Yiqun Wang, Yangjie Xu, Cedric Lothritz, Niccolo Gentile, Radu State, Tegawende F. Bissyande, and Jacques Klein. 2026. Do large language models grasp the grammar? evidence from grammar-book-guided probing in luxembourgish. *Preprint*, arXiv:2510.24856.

Shirong Ma, Yinghui Li, Rongyi Sun, Qingyu Zhou, Shulin Huang, Ding Zhang, Li Yangning, Ruiyang Liu, Zhongli Li, Yunbo Cao, Haitao Zheng, and Ying Shen. 2022. Linguistic rules-based corpus generation for native Chinese grammatical error correction. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 576–589, Abu Dhabi, United Arab Emirates. 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.

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.

Jakub Náplava, Milan Straka, Jana Straková, and Alexandr Rosen. 2022. Czech grammar error correction with a large and diverse corpus. *Transactions of the Association for Computational Linguistics*, 10:452–467.

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, Seattle, WA, USA → Online. Association for Computational Linguistics.

Frank Palma Gomez, Alla Rozovskaya, and Dan Roth. 2023. A low-resource approach to the grammatical error correction of Ukrainian. In *Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP)*, pages 114–120, Dubrovnik, Croatia. Association for Computational Linguistics.

David Peter. 2023. hyperfine: A command-line benchmarking tool. https://github.com/sharkdp/hyperfine.

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

Mengyang Qiu, Tran Minh Nguyen, Zihao Huang, Zelong Li, Yang Gu, Qingyu Gao, Siliang Liu, and Jungyeul Park. 2025. Multilingual grammatical error annotation: Combining language-agnostic framework with language-specific flexibility. In *Proceedings of the 20th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2025)*.

Qwen Team. 2025. Qwen3.5 technical report. https://qwenlm.github.io/blog/qwen3.5/.

D. E. Rozental'. 1997. *Spravochnik po pravopisaniyu i stilistike*. IC "Rossiyskaya gazeta", Moskva. [Handbook of spelling and stylistics].

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.

Sber AI. 2026. GigaChat 3.1 lightning. https://huggingface.co/ai-sage/GigaChat3.1-10B-A1.8B-bf16.

Tatiana Shavrina and Olga Shapovalova. 2017. To the methodology of corpus construction for machine learning: "Taiga" syntax tree corpus and parser. In *Proceedings of the International Conference "Corpus Linguistics–2017"*, Saint Petersburg.

Alexey Sorokin and Regina Nasyrova. 2025. LLMs in alliance with edit-based models: advancing in-context learning for grammatical error correction by specific example selection. In *Proceedings of the 20th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2025)*, pages 517–534, Vienna, Austria. Association for Computational Linguistics.

Felix Stahlberg and Shankar Kumar. 2021. Synthetic data generation for grammatical error correction with tagged corruption models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 37–47, Online. Association for Computational Linguistics.

Felix Stahlberg and Shankar Kumar. 2024. Synthetic data generation for low-resource grammatical error correction with tagged corruption models. In *Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024)*, pages 11–16, Mexico City, Mexico. Association for Computational Linguistics.

Swiss AI. 2025. Apertus-8b-instruct-2509. https://huggingface.co/swiss-ai/Apertus-8B-Instruct-2509.

Ekaterina Taktasheva, Maxim Bazhukov, Kirill Koncha, Alena Fenogenova, Ekaterina Artemova, and Vladislav Mikhailov. 2024. RuBLiMP: Russian benchmark of linguistic minimal pairs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9268–9299, Miami, Florida, USA. Association for Computational Linguistics.

Unsloth AI. 2024. Unsloth: Efficient LLM fine-tuning. https://github.com/unslothai/unsloth.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35.

An Yang, Baosong Yang, Beichen Zhang, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

A. A. Zaliznyak. 2010. *Grammaticheskij slovar' russkogo yazyka: Slovoizmenenie*, 6th ed. edition. AST-PRESS KNIGA, Moscow. [Grammatical Dictionary of the Russian Language: Inflection].

A Prompts

All finetuned models and baseline zero-shot evaluations use the system prompt proposed for LORuGEC evaluation:

> Дорогая языковая модель, после «Исходное предложение» тебе будет дано предложение на русском языке, которое может содержать орфографические, пунктуационные, грамматические и речевые ошибки. Выведи, пожалуйста, только корректный вариант данного предложения, не давая

никаких комментариев и не выделяя никаких символов. Твоя задача — минимально изменить текст, не меняй слова и знаки препинания, которые и так правильные.

The user message template is: Исходное предложение: `{src}`

Strict prompt. For the prompt sensitivity experiment (Table 7), we use the following system prompt:

> Исправь только знаки препинания и орфографию. Верни то же самое предложение, изменив только ошибочные запятые, тире, точки, дефисы или буквы, или их отсутствие. НЕ МЕНЯЙ СЛОВА. Каждое слово в ответе должно присутствовать в исходном предложении. Выведи только результат.

With bare `{src}` as the user message (no prefix).

B Training Details

Hardware. Small models ($\leq$7B) were trained on 4$\times$ RTX 5090 (vast.ai). Large models (8–12B) were trained on a single RTX 6000 Ada (vast.ai). Evaluation was performed on a DGX Spark (GB10 Blackwell, 128GB unified memory).

LoRA configuration. All models use LoRA with r=16, α=32, dropout 0, targeting all attention projections (q, k, v, o) and MLP projections $(gate, up, down)$. Training uses Unsloth (Unsloth AI, 2024) with full 16-bit base weights (LoRA, not QLoRA), effective batch size 32 (per-device 4 $\times$ gradient accumulation 8), learning rate 10^{-4} with cosine schedule. Early stopping monitors evaluation loss on LORuGEC validation (348 examples) with patience 5 epochs. Base training uses seed 42; continuation training uses seed 43.

Data. SyntErr training set: 39,209 examples across 48 LORuGEC rules (59 generation rules including bidirectional variants). Source text composition: 37% scarce-rule-enriched sentences (mined to boost underrepresented rules), 38% RuBLiMP scoring pool (Taktasheva et al., 2024), 25% news articles. LORuGEC training split: 348 examples (validation set of LORuGEC).

Models. Qwen3.5 (0.8B, 4B, 9B) (Qwen Team, 2025), Qwen2.5-7B (Yang et al., 2024), Gemma 3 (1B, 4B, 12B) (Gemma Team et al., 2025), Apertus-8B (Swiss AI, 2025), GigaChat-3.1-Lightning (Sber AI, 2026) (zero-shot only).

Inference. All local inference uses vLLM (Kwon et al., 2023) with bf16 precision, prefix caching enabled, and `max-model-len` 4096. Gemma 3 and Qwen 3.5 models require the `-language-model-only` flag (multimodal backbone with text-only inference). GigaChat-3.1-Lightning (Sber AI, 2026) uses the MLA attention backend with unquantized MoE routing. Frontier API models (GigaChat-2-Max, GPT-5.4, Claude Sonnet 4.6) were evaluated via their respective APIs. All models are evaluated in non-reasoning mode (`-no-think`): the model produces the corrected sentence directly, with no chain-of-thought or reasoning tokens.

Merge and deployment. LoRA adapters are merged into full-precision checkpoints using PEFT (Hugging Face, 2023) before evaluation. Qwen 3.5 models require `AutoModelForImageTextToText` for correct merging (multimodal architecture); Gemma 3 requires removal of `rope_parameters` from nested config levels for vLLM compatibility.

Reproducibility. All training and evaluation code, adapter checkpoints, configuration files, the SyntErr data generation pipeline, and the 39K-example SyntErr training set itself are released at `https://synterr-nlp.github.io/papers/bea-2026/`. The SyntErr source repository is at `https://github.com/synterr-nlp/synterr`. Training is deterministic given the seed, hardware, and library versions.

C Full Results

Table 10 reports full precision, recall, and $F_{0.5}$ for all models and settings on LORuGEC and GERA. Table 9 reports RuCoLA grammaticality judgment (MCC).

Model	Par.	ZS	+lor	+syn	+s→l
Qwen3.5-0.8B	0.8B	.291	.262	.166	.239
Qwen3.5-4B	4B	.460	.472	.477	.473
Qwen3.5-9B	9B	.522	.521	.521	.532
Qwen2.5-7B	7B	.493	.389	.412	.398
Gemma-3-1B	1B	.202	.211	.119	.117
Gemma-3-4B	4B	.467	.471	.414	.445
Gemma-3-12B	12B	.501	.497	.488	.520
Apertus-8B	8B	.436	.428	.479	.457

Table 9: RuCoLA OOD dev MCC (1804 examples). 0 = chance, 1 = perfect. Models $\geq$4B: stable; smaller models degrade under SyntErr.

(a) LORuGEC

Model	Par.	Zero-shot			+lorugec			+synterr			+s → l		
		P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Qwen3.5-0.8B	0.8B	21.2	5.7	13.7	45.0	16.9	33.8	60.4	22.6	45.2	60.7	37.5	**54.0**
Qwen3.5-4B	4B	52.3	34.9	47.6	59.8	40.8	54.7	79.7	40.9	67.0	81.8	57.1	**75.3**
Qwen3.5-9B	9B	51.9	40.8	49.2	65.4	48.7	61.2	62.0	44.1	57.3	74.1	60.3	**70.9**
Qwen2.5-7B	7B	42.0	31.9	39.5	49.4	30.6	44.0	70.6	34.0	58.1	72.3	50.6	**66.6**
Gemma-3-1B	1B	21.7	12.5	18.9	27.1	20.8	25.5	58.6	27.9	**48.0**	48.4	45.5	47.8
Gemma-3-4B	4B	43.8	34.3	41.5	43.2	42.0	43.0	59.3	41.0	54.5	59.2	54.5	**58.2**
Gemma-3-12B	12B	54.9	44.7	52.5	60.6	52.7	58.8	71.1	51.5	66.0	71.2	64.2	**69.7**
Apertus-8B	8B	51.5	39.9	48.7	66.0	31.5	54.1	78.4	47.4	69.4	76.8	58.8	**72.4**
GC-3.1*	10B	60.7	33.1	52.0	—	—	—	—	—	—	—	—	—

(b) GERA (out-of-domain for SyntErr)

Model	Par.	Zero-shot			+lorugec			+synterr			+s → l		
		P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Qwen3.5-0.8B	0.8B	32.8	13.9	25.8	48.6	20.3	38.0	66.2	18.8	**44.0**	43.4	23.1	36.9
Qwen3.5-4B	4B	59.0	48.4	56.5	52.0	35.0	47.4	62.4	24.9	47.9	71.8	41.9	**62.8**
Qwen3.5-9B	9B	56.5	58.1	56.8	58.1	43.1	54.4	57.8	38.1	52.4	62.5	49.7	**59.4**
Qwen2.5-7B	7B	48.3	44.9	47.6	47.4	35.6	44.5	57.7	24.7	45.5	56.7	35.2	**50.5**
Gemma-3-1B	1B	27.8	22.8	26.6	27.6	27.4	27.6	50.2	21.0	**39.3**	32.6	30.7	32.2
Gemma-3-4B	4B	47.3	42.7	46.3	41.2	46.9	42.3	54.5	34.7	**48.9**	47.3	44.1	46.7
Gemma-3-12B	12B	53.6	54.6	53.8	55.2	55.8	55.4	65.3	45.2	**59.9**	58.7	54.8	57.9
Apertus-8B	8B	52.1	50.0	51.7	65.1	33.7	54.9	72.6	37.5	61.1	68.2	45.7	**62.1**
GC-3.1*	10B	65.7	45.5	60.4	—	—	—	—	—	—	—	—	—

Table 10: Full P, R, $F_{0.5}$ on (a) LORuGEC and (b) GERA (M2scorer, ë/dash normalized). *MoE: 10B total, 1.8B active.

D SyntErr Generation Details

Architecture. SyntErr is built around three core abstractions: `ErrorHandler` (a protocol from which all rule-specific error generators inherit), `Analyzer` (the morphological/syntactic backend), and `ErrorPipeline` (which orchestrates distribution sampling, candidate selection, and corruption). Rules are classified as *length-preserving* (e.g., vowel alternations, suffix errors, paronym substitution) or *length-changing* (e.g., function word insertion/deletion, compound-form splitting/merging). After any length-changing rule fires, `ErrorPipeline` recalculates all downstream token indices to keep the alignment between source and target consistent. The full pipeline is shown in Figure 2.

Rule examples. Consider, for example, the algorithm for introducing a "paronym" type error. First, the word is looked up in a paronym dictionary. If the word is found, it is replaced with one of its paronyms, selected at random. The new word is then inflected using `pymorphy3` to preserve the original grammatical categories.

Among native speakers of Russian, spelling errors involving the use of н and нн in suffixes are quite common. To simulate the error of writing н instead of нн, we utilize a morpheme dictionary to verify that нн constitutes part of the suffix rather than belonging to the root or appearing at a morpheme boundary. We then delete one н. Conversely, to generate the opposite error, we verify that н belongs to one of the suffixes that take a single н (such as -ан-, -ян-, -ин-) and that the target word is not among the exceptions, after which we add an additional н.

A more complex example involves comma placement errors (insertion of an extraneous comma). According to the rules outlined in Rozental's reference guide, an extra comma is inserted in the following cases where it is not required:

1. A comma before the conjunction как when it means в качестве (*as*) or is part of idiom. Uses dep-tree: only targets как with dep_rel=case/fixed/flat (adjunct/apposition sense), NOT advcl/ccomp/mark (subordinate clause).

2. A comma within a phraseological unit containing repeated conjunctions (e.g. ни слуху

ни духу).

3. A comma between adjacent conjunctions at clause boundaries. Only fires when a то/так/но correlative follows the subordinate clause.

4. A comma within indivisible expressions (e.g. как ни в чём ни бывало, куда глаза глядят).

Examples of sentences with errors of these types generated using SyntErr are presented in Table 11.

Performance. SyntErr generates 10K examples in under 2 minutes on CPU; a GPU is not required but significantly accelerates stanza, especially when using depparse.

We measured generation time on a laptop (AMD Ryzen 5 7535HS, 6 cores; NVIDIA RTX 4050 6 GB; 16 GB RAM; Ubuntu 24.04 LTS) using a mixed source dataset compiled from RuBLiMP (Taktasheva et al., 2024), Taiga (Shavrina and Shapovalova, 2017), Russian Wikipedia,[4] and news articles.[5] Benchmark presets use error probability 1.0 and max one error per sentence; `profile_*` presets isolate handler categories.

Table 12 compares backends and depparse overhead. Natasha is fastest but lacks dependency parsing, limiting it to spelling and morphological rules. Stanza is roughly 5× slower on CPU than on GPU (Table 12); GPU throughput is roughly independent of batch size above 64 (Table 14). Per-category timings (Table 13) show uniform throughput across error types.

Rule coverage. The 48 LORuGEC rules expand to 59 generation rules (bidirectional variants for split/merge and comma insertion/deletion). Each generation rule was assigned a uniform target of 847 examples; 40 rules reached this target, and the remaining 19 are limited by source sentence availability. Punctuation rules are directionally skewed toward comma insertion (3.6:1 comma insertion skew).

Dependency-driven error generation. The key design choice is that most error types are applied at positions identified through dependency parsing, not at random. Three families of errors rely on the parse tree:

Punctuation (§§77–138): comma errors inspect clause boundaries, conjunction scopes, and modifier attachment. For compound sentences (§87), SyntErr finds the `conj` arc between predicates joined by и/а/но and deletes the required comma. Spurious insertion follows the same logic in reverse: inserting commas where they are *not* required (before как in non-comparative contexts, inside phraseological expressions, between adjacent conjunctions).

Agreement (§§191–197): adjective-noun errors traverse `amod` arcs; subject-verb errors use `nsubj` arcs.

Government (§§196–210): prepositional government errors corrupt case via `obl`/`nmod` arcs.

Rules not requiring syntax — spelling (§§1–34), paronym substitution, compound splitting — use morphological tags and dictionary lookup. Inflection uses pymorphy3; stress-sensitive rules use Zaliznyak's dictionary (Zaliznyak, 2010).

Source text filtering. Before generating errors, source sentences are filtered to keep those between 10 and 50 tokens (long enough to host a target rule, short enough to validate by hand). They are then deduplicated to avoid repetition, and any sentence appearing in the RuBLiMP benchmark is removed to prevent evaluation-time leakage.

[4] https://dumps.wikimedia.org/

[5] https://github.com/mannefedov/ru_kw_eval_datasets

Original	Corrupted	Error	Fix tag
Отправляясь на охоту , он надел ветровку болотного цвета .	Отправляясь на охоту , он одел ветровку болотного цвета .	paronym @ position 5	$REPLACE_надел
Шум прибоя растёт , осенний ледяной ветер вздымает и бешено срывает волны , разнося по воздуху брызги и резкий запах моря .	Шум прибоя растёт , осенний ледянной ветер вздымает и бешено срывает волны , разнося по воздуху брызги и резкий запах моря .	orthographic_spelling_nn_suffix @ position 27	$REPLACE_ледяной
Утверждают , что бразильские карнавалы восхищают и завораживают , и когда мы впервые увидели его неповторимую яркую красоту , то сами убедились , насколько правы были очевидцы .	Утверждают , что бразильские карнавалы восхищают и завораживают , и , когда мы впервые увидели его неповторимую яркую красоту , то сами убедились , насколько правы были очевидцы .	comma_insert_ comma_between_conjunctions @ position 10	$DELETE

Table 11: Examples of SyntErr-generated errors

Backend	Dep.	500	1K	2K	10K
stanza (GPU)					
+ depparse	yes	11.4	14.4	20.7	71.6
no depparse	no	9.9	12.1	16.4	51.1
stanza (CPU)					
+ depparse	yes	28.9	45.9	77.1	348
no depparse	no	15.9	28.2	44.4	190
natasha	no	3.4	5.4	8.7	37.3
spaCy	no	7.0	9.2	14.0	50.9

Table 12: SyntErr generation time in seconds (balanced preset; mean over 5 runs, measured with hyperfine (Peter, 2023)). lorugec preset within 5% of balanced.

Table 13: Preset performance (2K sentences, stanza GPU+depparse). S/s = sentences per second.

Preset	Time (s)	S/s	Notes
profile_spell	19.6 ± 0.2	102	no inflection
profile_morph	19.6 ± 0.1	102	pymorphy3
profile_punct	19.8 ± 0.0	101	dep traversal
profile_struct	19.5 ± 0.1	102	ins/del tokens
balanced	20.4 ± 0.7	98	reference

Table 14: Batch size vs. throughput (2K sentences). S/s = sentences per second.

Batch	Time (s)	S/s	GPU MiB
32	22.0 ± 0.1	91	1059
64	20.6 ± 0.3	97	1443
128	20.0 ± 0.1	100	1975
256	19.8 ± 0.1	101	3443
512	19.4 ± 0.2	103	3541

In force-apply mode (all 48 rules, bidirectional), stanza GPU processes 10K sentences in 79 s. Peak memory: 1.8 GB (stanza GPU+depparse), 0.4 GB (natasha), 1.1 GB (spaCy).

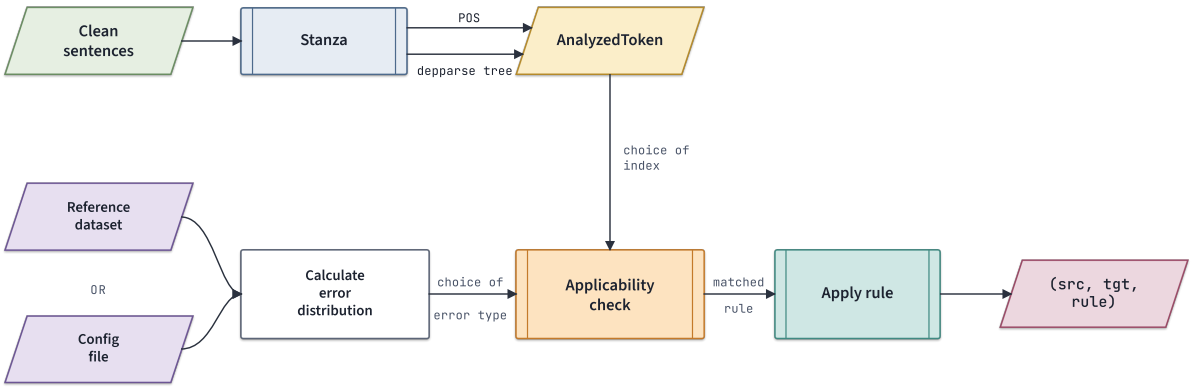

Figure 2: SyntErr error generation pipeline. Clean sentences are parsed by Stanza into analyzed tokens; a per-rule error-type distribution (from a reference corpus or a config file) drives an applicability check that selects a matched rule; the rule is then applied to produce a `(src, tgt, rule)` triple.

E Scoring Normalization

Before scoring, we apply two normalizations. First, ё is mapped to е: LORuGEC targets use е consistently, but models frequently produce the formally correct ё. Second, Unicode dash variants (FIGURE DASH U+2012, EN DASH U+2013) are mapped to EM DASH U+2014; the gold data uses these inconsistently (56 em-dash vs. 6 en-dash in LORuGEC). Hyphen-minus is not normalized, as it is semantically distinct. We do not normalize quotation marks: Russian prescriptive typography requires angle quotes («ёлочки»), but some gold examples use straight quotes. Models producing the correct form are penalized; we accept this small bias rather than erase a genuine distinction.