

Getting More from Small LLMs and Limited Data: Fuzzy-Match Retrieval and Candidate Scoring for Multitask Sorbian NLP

Thomas Moerman and Arda Tezcan

LT³, Language and Translation Technology Team, Ghent University

Groot-Brittanniëlaan 45, 9000 Ghent, Belgium

{thomas.moerman, arda.tezcan}@ugent.be

Abstract

We describe LT3’s submission to the WMT26 shared task on Multitask LLMs with Limited Resources for Upper Sorbian (hsb) and Lower Sorbian (dsb): one QLoRA-fine-tuned Qwen3.5-2B model serves machine translation, multiple-choice question answering, spell checking, grammar checking, and maths reasoning. Our approach centres on three components. First, *retrieval augmentation*: fuzzy-match (FM) exemplars, retrieved by character- and embedding-based similarity from authentic and back-translated sentence pools, are included in prompts at training and inference times. Second, *synthetic task data*: training data for spell- and grammar-checking is generated from dictionaries and monolingual text. Third, a *scoring-based inference system*: the model generates free text only for the open-ended tasks—translation and maths reasoning—while, for the other three tasks, it scores candidates from constrained spaces by ranking multiple-choice options in place and selecting spelling and grammar corrections from dictionary-derived candidates. Our primary system was the joint winner of the Sorbian track, ranking first or second across all five tasks and outperforming the next-best system by 7.7 accuracy points on question answering and 1.4 points on mathematical reasoning. These results show that, for low-resource languages, FM augmentation combined with back-translation can improve MT performance for a relatively small LLM, while inference-level strategies can further improve its performance on other tasks without compromising MT performance.

 [Code](#)

 [Dataset · Models](#)

1 Introduction

Upper Sorbian (hsb) and Lower Sorbian (dsb) are West Slavic minority languages with scarce digital resources, and the WMT26 shared task on Multi-

task LLMs with Limited Resources (Okabe et al., 2026) asks a single model of at most two billion parameters to serve five tasks in both languages: machine translation (MT), multiple-choice question answering (QA), spell checking (SC), grammar checking (GC), and mathematical reasoning (MR). For MT, German additionally serves as both source and target language, yielding six translation directions in total.

Our approach centres on three components. The first is **retrieval augmentation**: fuzzy-match (FM) exemplars, retrieved by character- and embedding-based similarity, are included in prompts at both training and inference. This component is motivated by prior work showing that FM augmentation improves neural MT (Bulté and Tezcan, 2019), with further gains in translation performance obtained when back-translation (BT) enlarges the pool from which FMs are retrieved (Tezcan et al., 2024), and that these benefits carry over to fine-tuned LLMs (Moerman et al., 2026). We bring this approach to a multitask LLM setting in new language directions, while changing the role of BT: fine-tuning uses only authentic sentence pairs for FM retrieval, whereas back-translated sentence pairs are also used at inference (Section 4).

The second component is **synthetic task data**: none of the four non-MT tasks comes with task-formatted training data, so we generate it, turning dictionaries and monolingual text into spell- and grammar-checking training pairs and building QA and MR items from monolingual sentences and translated seed problems (Section 5).

The third component is a **scoring-based inference system**. Only MT and MR require the model to write free text. For QA, SC, and GC it uses scoring instead. A deterministic layer lays out the possible answers, QA options in their question context, and SC and GC corrections drawn from dictionaries, and the model picks the answer to which it assigns the highest log-likelihood in place (Sec-

tion 6).

Our primary system was the joint winner of the Sorbian track, ranking first on QA (65.05, +7.72 accuracy points over the next-best system) and outright on MR (30.40, +1.40), while placing second on the three remaining tasks.

Section 2 introduces the shared task setup. Section 4 describes our MT approach, covering data preparation, fine-tuning, and exemplar retrieval. Section 5 then details how the remaining four tasks are incorporated into training, while Section 6 presents the inference strategies applied after training.

2 Shared Task Setup

The WMT26 shared task on Multitask LLMs with Limited Resources (Okabe et al., 2026), continuing the WMT25 series (Okabe et al., 2025), asks one model to serve five tasks in a low-resource setting. We participated in the Sorbian track, covering Upper Sorbian (hsb) and Lower Sorbian (dsb), two West Slavic minority languages. Table 1 lists the five tasks with their output spaces, hidden-test sizes, and official metrics. MT, the only task involving German, runs over six directions (between German, Upper and Lower Sorbian, in both ways), and its hidden-test size is summed over the directions. Per-task scores average the hsb and dsb subtasks. Models must come from the Qwen 3.5 family (Qwen Team, 2026) with at most 2B parameters, the same model must serve all five tasks within a submission, training data must be publicly available, and retrieval augmentation from a clearly defined corpus is permitted at both training and inference. We refer to the task page¹ and the organisers’ repository² for the full task, data, and rule details. The fine-tuned models we submitted are publicly released³ together with our code and data (Appendix D).

3 Related Work

We highlight four lines of prior work that are utilised in our submission: (i) shared-task efforts around the Sorbian languages, (ii) retrieval augmentation for translation, (iii) language models as scorers of constrained candidates, and (iv) the training dynamics of multitask fine-tuning.

¹<https://www2.statmt.org/wmt26/limited-resources-llm.html>

²<https://github.com/TUM-NLP/llms-limited-resources2026>

³<https://huggingface.co/TomMoeras/wmt26-lt3-sorbian>

Task	Output space	Test	Official metric
MT	free text	26,174	chrF++
QA	one option per slot	415	accuracy
SC	flag + correction	4,000	mean det./corr. EM
GC	flag + correction	3,726	mean det./corr. EM
MR	numeric answer	500	accuracy

Table 1: The five Sorbian-track tasks. Test = hidden-test items (MT summed over six language directions, the remaining four tasks summed over hsb and dsb). SC and GC are scored as the mean of detection exact match and correction exact match, so a system must both flag erroneous sentences and reproduce the corrected sentence exactly.

Sorbian and low-resource shared tasks. The task setting itself has a history at WMT. Upper and Lower Sorbian entered the evaluations through the unsupervised and very low resource MT tasks (Fraser, 2020; Libovický and Fraser, 2021; Weller-Di Marco and Fraser, 2022), and the step from pure translation to a multitask LLM came with WMT25’s joint MT and QA track (Okabe et al., 2025), whose participants adapted Qwen2.5 models with external data and varying fine-tuning strategies (Purason and Fishel, 2025; Larkin et al., 2025; Saadi et al., 2025). Those findings also record a first interaction between the joint tasks: training purely on MT degrades QA. We revisit that interaction along a different axis, training duration within one fixed mixture (Section 5).

Retrieval-augmented translation. On the translation side, prior work asks how to make better use of scarce parallel data. One answer is retrieval-augmented translation: FM augmentation concatenates retrieved translation-memory matches to the source and improves NMT (Bulté and Tezcan, 2019), embedding-based retrieval extends the idea beyond edit distance (Xu et al., 2020), and the interaction between source and retrieved match has been studied for domain adaptation, showing that controlling how strongly the model relies on the retrieved match yields further gains (Hoang et al., 2023). The same idea reappears with LLMs as exemplar selection for translation prompting (Agrawal et al., 2023; Vilar et al., 2023), including FMs in adaptive translation (Moslem et al., 2023). A second answer is to generate synthetic data: back-translation (Sennrich et al., 2016) converts monolingual text in the target language into synthetic source-target pairs, although Edunov et al. (2018) show that using such pairs for training can be detrimental in some settings. The two lines of work

converge in [Tezcan et al. \(2024\)](#), who combine FM augmentation with back-translation for NMT, and [Moerman et al. \(2026\)](#), who extend this combination to LLMs. We build on this combination but split it by stage: authentic sentence pairs serve FM retrieval during training, while synthetic pairs obtained through back-translation are added to the pool for retrieval at inference, and we measure each stage’s effect (Section 4). To our knowledge, neither this separation nor the role of exemplar provenance in training stability has been studied before.

Scoring rather than generating. The non-generation tasks draw on a different tradition, in which a language model scores constrained candidates rather than generating freely. For multiple choice, scoring the likelihood of each option is the standard evaluation convention ([Brown et al., 2020](#)), with known calibration caveats ([Holtzman et al., 2021](#); [Zhao et al., 2021](#)), and when several questions share one option bank, the choices become a joint-assignment problem, which SCDE ([Kong et al., 2020](#)) poses for sentence cloze and solves by exhaustive enumeration. The same generate-candidates-then-score pattern has a long history in spelling correction, where a candidate generator proposes, and a language model decides in context ([Kernighan et al., 1990](#); [Brill and Moore, 2000](#)), a recipe carried to Slavic spelling and diacritics by Korektor ([Richter et al., 2012](#)). For grammatical error correction, synthetic corruptions substitute for scarce annotated data ([Náplava and Straka, 2019](#)), and the acceptability judgement itself can come from a language model ([Yasunaga et al., 2021](#)). Even translation decoding has used the same principle, re-scoring hypotheses with a noisy-channel model ([Yee et al., 2019](#)). In each of these lines, the scorer is built separately for its task. In our system, a single fine-tuned model plays that role for QA, SC and GC at once (Section 6). We also depart from SCDE in how we find the assignment, computing the best one directly ([Kuhn, 1955](#)) instead of trying every combination, and we accept a correction only when real corpus text attests it.

Fine-tuning dynamics and checkpoint selection. Finally, packing several tasks into one set of weights raises the question of when to stop training. [Koh et al. \(2026\)](#) document heterogeneous per-task overfitting in multitask supervised fine-tuning (SFT): under one shared budget, sub-datasets reach their optima at different steps. Selecting the stop-

ping point by validation loss has its own pitfall, as [Isik et al. \(2025\)](#) show that held-out cross-entropy can keep improving monotonically while translation quality worsens. Both observations appear in our training trajectories and informed how we selected final checkpoints (Section 5).

4 Machine Translation

We first developed the MT component and subsequently built the remaining system components around it. A working MT engine serves the six required translation directions directly, but it also allows us to generate back-translations that enlarge the pool of sentences used for retrieval-based MT, and to translate the English seed problems to generate additional MR training data for Upper and Lower Sorbian languages (see Section 5).

4.1 Parallel Data and Back-Translation

The authentic parallel data combines the organiser-provided datasets with public German–Sorbian corpora from earlier WMT editions: the German–Upper Sorbian sets from WMT2020–2022 and the German–Lower Sorbian and Upper–Lower Sorbian sets from WMT2022. Although the available parallel corpora are broadly comparable across language pairs, we observed a clear asymmetry in translation performance: for both the zero-shot base model and models fine-tuned on the available (authentic) parallel datasets, performance was substantially lower when translating from German into either Sorbian variety.

To improve these lower-performing German-source directions, we back-translated additional monolingual Sorbian data provided by the organisers and drawn from public Sorbian corpora into German ([Sennrich et al., 2016](#)), using the corresponding fine-tuned Sorbian-source models. This yielded additional synthetic German–Sorbian sentence pairs for the de→hsb and de→dsb directions.

Table 2 summarises the total number of authentic sentence pairs for each translation direction, the additional synthetic sentence pairs generated through back-translation, and the chrF++ scores of both the zero-shot base model and the fine-tuned models on the gold development sets.

4.2 Fuzzy-Match Retrieval

In FM augmentation, the model retrieves translations of source sentences similar to the input (fuzzy matches, FMs) from a translation memory (TM)

Direction	MT data (pairs)		Total	chrF++	
	Authentic	Synthetic		Base	FT
de→hsb	261,764	500,000	761,764	7.66	53.60
de→dsb	113,680	165,738	279,418	7.58	49.37
hsb→de	261,764	–	261,764	30.77	62.54
dsb→de	113,680	–	113,680	25.34	59.57
hsb→dsb	159,193	–	159,193	40.65	73.91
dsb→hsb	159,193	–	159,193	42.92	73.98
total	1,069,274	665,738	1,735,012		

Table 2: The overview of the data sets and the MT performance. Number of authentic sentence pairs, synthetic sentence pairs and total number of sentence pairs per language direction, combined with the chrF++ scores, on the gold development sets, of the zero-shot base model (Base) and of the fine-tuned (FT) models that utilise the authentic datasets.

and leverages them alongside the input to produce MT output. The technique improves neural MT performance (Bulté and Tezcan, 2019), with further gains when the pool from which FMs are retrieved is expanded through back-translation (Tezcan et al., 2024), and has also been shown to improve translation performance in fine-tuned LLMs (Moerman et al., 2026). We build directly on this line and extend it to a multitask LLM setting, with one adjustment: Moerman et al. (2026) used the back-translated pairs during fine-tuning as well, whereas in this work, fine-tuning relies entirely on authentic sentence pairs, while back-translated pairs additionally serve FM retrieval only at inference. The noisier synthetic pairs therefore never reach training, but still provide additional value at inference: using them to expand the FM retrieval pool yields a modest further improvement in chrF scores (see Table 3). In the **training stage**, we sample the number of in-context exemplars per row uniformly from $\{0, 1\}$: a seeded 50% of MT training rows carry exactly one top-1 FM retrieved (only) from the authentic pairs (one-shot), and the remaining rows carry none (zero-shot). This uniform-sampling (0/1) scheme was the most effective configuration in Moerman et al. (2026), drawing on the uniform shot-count sampling of Alves et al. (2023). It differs from the **inference stage**, where every test source retrieves $k = 3$ FMs from the full authentic+BT pool. Retrieval ranks candidates by cosine similarity of MiniLM sentence embeddings (Wang et al., 2020) with FAISS indexing (Johnson et al., 2019) for the German-source directions, and, because the embedding model does not cover Sorbian, by character-set Jaccard similarity for the Sorbian-

Training stage	0 or 1 exemplar · authentic pool only
Translate the source text from German to Upper Sorbian	
German: Die Kinder und die Eltern haben das Sprachangebot gut angenommen.	real · .971
Upper Sorbian: Džěći a starši su rěčny poskitk derje přiwzali.	
German: So erleben die Kinder Freude mit und an der Sprache.	
Upper Sorbian:	
Inference stage	k=3 exemplars · authentic+BT pool (600k)
Translate the source text from German to Upper Sorbian	
German: Sie lernen aber auch mit der Geschichte der Kunst.	BT · .973
Upper Sorbian: Zeznajomjeja pak so tež ze stawiznami wuměłstwa.	
German: So konnten sie sich besser auf das Erlebnis und die Gemeinschaft konzentrieren.	BT · .972
Upper Sorbian: Tak móžachu so lěpje na dožiwjenje a na zhromadnosć koncentrować.	
German: Ihre Freude an der Gesang und dem Spielen wird schnell auf das Publikum übertragen.	BT · .972
Upper Sorbian: Jich radosć ze spěwanja kaž tež hudźenja so spěšnje na publikum přenese.	
German: So erleben die Kinder Freude mit und an der Sprache.	
Upper Sorbian:	

Figure 1: The two retrieval stages on the same deu→hsb development-set source (bold). During training, under uniform sampling (0/1), 50% of the training rows are augmented with one FM retrieved from the authentic TM (real), while the remaining 50% contain no FMs (zero-shot). At inference, we retrieve three FMs from the full authentic+BT TM. For this source, all three highest-similarity FMs originated from back-translated (BT) pairs, where the corresponding cosine similarity score is shown for each FM.

source directions. Figure 1 walks one example through both stages. The example also illustrates why the enlarged retrieval pool by the synthetic sentence pairs may lead to improvements in translation performance: all three of the retrieved exemplars at inference are synthetic BT pairs, close neighbours that are markedly more similar to the source than anything the authentic-only pool of the training stage offers. Back-translated pairs exist only for the two German-source directions, and there they dominate retrieval: 65.9% of the exemplars retrieved for de→hsb and de→dsb are synthetic BT (72.4% and 60.0% respectively). The four Sorbian-source directions are retrieved from authentic-only pools, so no synthetic pair is ever retrieved for them. Fine-tuning uses none. We validate both design choices at the end of this section.

4.3 MT Fine-Tuning

The MT training set contains 1,069,344 rows of authentic parallel data only, rendered under the uniform-sampling (0/1) scheme above (one seeded

carrier mask fixes which rows carry an FM). We fine-tune Qwen3.5-2B with QLoRA (Detrmers et al., 2023) at rank 256 (full training configuration in Appendix B).

4.4 Validating the Configuration

The choices above, fuzzy rather than random carriers and BT kept out of the fine-tuning rows, were validated on the gold development sets (Table 3). On the base model, in-context FMs used only during inference (not during fine-tuning) reduce MT performance: averaged over all language directions, zero-shot scores 25.82 chrF++, three random exemplars 24.74, and three FMs 24.61. We hypothesise that this is an effect of model scale: in-context learning is known to be effective for MT specifically (Moslem et al., 2023; Moerman et al., 2026), but with larger models, and few-shot ability in general strengthens with model size (Brown et al., 2020). The improvements that follow therefore rely on combining fine-tuning with in-context exemplars, together lifting MT performance by nearly fifty chrF++ points. We compare four fine-tuning recipes, crossing the carrier exemplar selection (fuzzy versus random) with the source of the fine-tuning rows (authentic only versus authentic+BT), and evaluate each recipe at its converged optimum under four inference settings that cross the same two choices on the retrieval side. Figure 2 traces the four recipes across the training schedule. Training to convergence separates them: the recipe of our primary system, fuzzy carriers drawn from authentic rows, improves progressively to a 76.05 plateau at 48k, while the other three peak by 12–15k and then degrade. A plausible reading is that fuzzy carriers keep the exemplar informative, so the copy-and-adapt behaviour they teach keeps improving with more training, whereas random exemplars and synthetic German pairs act as noise that longer training increasingly overfits to. The main improvement thus comes from fuzzy carriers during training, drawn from authentic rows only: at convergence the fuzzy-over-random margin is +3.63, and authentic-only rows end +6.26 ahead of authentic+BT rows under fuzzy carriers. The inference-side choices matter far less. Retrieving FMs rather than random exemplars adds 0.25 chrF++ on the converged primary recipe, and enlarging the retrieval pool with BT adds a further 0.06, which brings the deployed configuration to 76.11, the best cell of the grid. Notably, the random-trained recipes gain nothing from fuzzy retrieval at

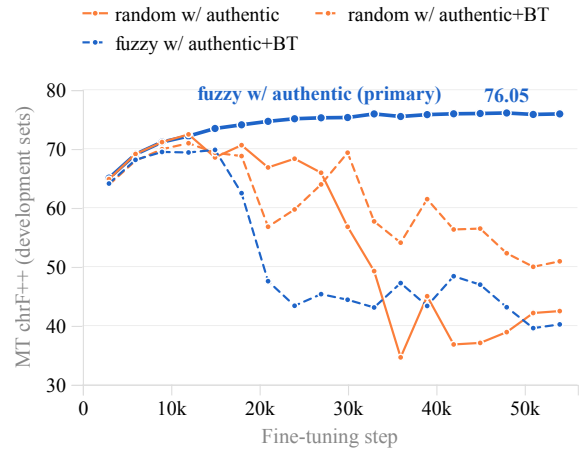


Figure 2: Development-set MT chrF++ of the four fine-tuning recipes averaged over de→hsb and de→dsb (carrier exemplar selection w/ fine-tuning row source), all trained under uniform sampling (0/1). All four curves use the same inference at every checkpoint, $k = 3$ FMs retrieved from the authentic pool, so only the fine-tuning recipe differs. Markers show the evaluated checkpoints. Only the chosen configuration converges upward to a 76.05 plateau, while the other three peak by 12–15k steps and degrade under further training.

inference, so the ability to exploit retrieved exemplars is itself learned during fine-tuning.

5 Folding In the Other Tasks

With the MT engine in place, the remaining four tasks are folded into the same model via task-specific fine-tuning.

5.1 Synthetic Task Data

For the four non-MT tasks, no training data exists in the task format: the only task-formatted data, the organiser development sets, is reserved for evaluation and selection, and what remains is raw monolingual corpora. We therefore synthesised the training data, using a different strategy per task. For SC, character-level typo injection turns clean sentences into (erroneous, corrected) pairs. For GC, morphological paradigm-swap corruptions generated with the soblex (Baier et al., 2026) and dsb-spell (derbnik, 2024) lexicons play the same role, following the synthetic-corruption route of low-resource GEC (Náplava and Straka, 2019). For QA, cloze-style multiple-choice items are built from monolingual sentences. For MR, public English seed problems (GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), with PolyMath excluded per the track rules) were translated into Sorbian by the best-performing MT engine (Section 4), each paired with a worked so-

	Inference			
	fuzzy		random	
<i>Fine-tuning</i>	auth.	auth. +BT	auth.	auth. +BT
<i>no fine-tuning (zero-shot 25.82)</i>				
base model	24.61	–	24.74	–
<i>authentic</i>				
fuzzy	76.05	76.11	75.80	75.80
random	72.42	72.44	72.40	72.40
<i>authentic + BT</i>				
fuzzy	69.79	69.97	68.55	68.34
random	70.92	70.99	70.87	70.84

Table 3: Development-set MT chrF++, averaged over the six directions, for the base model and for each fine-tuning recipe (carrier selection within row source, evaluated at its converged optimum) under the four inference settings (exemplar selection within retrieval pool, $k = 3$ throughout).

Task	Original (dev. sets)	Synthetic	
		hsb	dsb
SC	4,000	100,000	15,000
GC	3,250	100,000	15,000
QA	158	60,000	12,000
MR	48	28,147	

Table 4: Number of items in the original versus synthesised task data for the four non-MT tasks (hsb and dsb combined). The organiser development sets are the only task-formatted data provided, and they are reserved for evaluation and selection, so all task-formatted training data is synthesised.

lution. The per-language volumes follow the available monolingual data, whose clean-sentence supply is roughly six times smaller for dsb than for hsb. Table 4 contrasts these synthetic volumes with the only original task-formatted data that exists, the organiser development sets.

5.2 The Multitask Mix

The base mix combines the 1,069,344 MT training rows of Section 4, the authentic per-direction rows of Table 2 summed over the six directions, with the synthetic data sets generated for the remaining four tasks, as provided in Table 4, upsampled to fixed shares: 1,397,057 rows in total, of which MT 76.5%, MR 7.0%, SC 6.3%, GC 6.3%, and QA 3.9%. One QLoRA run over this mix produces the base multitask model.

5.3 Three Fine-Tuning Phases

Fine-tuning proceeds in three phases (Table 5).

Phase 1: the multitask base model. Phase 1 trains the base model on the mix above. Tracking

all five tasks on the development sets along this run reveals a clear split (Table 6). Three tasks improve for as long as we train, up to our last training step (27,000): MT (+12.1 chrF++), SC (+.165) and GC (+.162), while QA peaks at 9,000 steps and slowly declines thereafter, and MR fluctuates on its small development slice. No single checkpoint is best for every task, so selection goes by the equal-weight average, which climbs to a broad optimum between 18,000 and 27,000 steps, and even QA’s early peak costs only .044 from inside that region. Appendix A plots the full trajectory.

Phase 2: GC continuation. Even at its phase-1 best, GC remains the weakest task, so phase 2 targets it: starting from the merged phase-1 weights, a short continuation trains on a mixture dominated by GC corruptions produced with the same generator as above, at larger volume (520k rows, 63.4% of an 820k-row mixture). The continuation converts the remaining GC headroom while holding the other tasks (Table 7): GC gains +.116 and saturates within 3,000 continuation steps, SC and MR drift up slightly, QA stays flat, and MT gives up half a chrF++ point. The development-set optimum lies at continuation step 3,500, on the GC plateau. In short, increasing the share of GC data in the training mixture improved GC further at little cost to the other tasks.

Phase 3: folding in the development sets. Phase 3 asks whether the scarce organiser development sets can themselves add value as training data: it adds the development sets to the phase-2 mixture (+31,614 rows, rendered in the official evaluation formats so that training examples match the inference format exactly), again starting from the merged phase-1 weights. Both continuations train adapters that merge onto the merged phase-1 weights. A development-folded model can no longer be evaluated or selected on the development sets it has seen, so its effect shows only on the hidden test, where it is worth +.002 aggregate (Section 8), and checkpoint selection needs dedicated protocols. We define three protocols and build three corresponding submission types around them (Section 7). **EarlyStop** picks the checkpoint minimising mean cross-entropy on a frozen held-out slice, using no development and no test signal. **DevTransfer** transfers the checkpoint step from the phase-2 sibling, which never saw the development sets and may therefore be selected on them. **Held-OutDev** simply stops at phase 2. The primary

Phase	Training mixture	Starts from
1 multitask base	1,397,057 rows	Qwen3.5-2B
2 GC continuation	300k + 520k GC corruptions (820k)	merged phase 1
3 development fold	phase-2 mixture + 31,614 devel. rows	merged phase 1

Table 5: The three fine-tuning phases. Phases 2 and 3 are QLoRA continuations whose adapters merge onto the merged phase-1 weights. Phase 3 differs from phase 2 only by folding the organiser development sets into the training mixture. Submissions run phase-3 weights except FullStack-HeldOutDev, which stops at phase 2 (Table 9).

Checkpoint	MT	QA	SC	GC	MR	eqw
3,000	62.54	.611	.556	.474	.208	.495
9,000	70.12	.620	.643	.545	.229	.548
15,000	72.53	.554	.682	.600	.188	.550
24,000	74.29	.576	.714	.630	.229	.578
27,000	74.65	.589	.721	.636	.188	.576

Table 6: Development-set scores along the phase-1 trajectory (best per column in bold), where eqw corresponds to the equal-weight average after the chrF++ scores for MT were divided by 100 for scale normalisation.

system, FullStack-DevTransfer, follows the DevTransfer protocol: it runs phase 3 for 3,500 continuation steps on top of the phase-1 checkpoint (each continuation counts its steps from zero), and transfers the step count from phase 2’s development-set optimum.

6 Inference Strategies

At inference, the frozen model plays two roles. For the open-ended tasks, MT and MR, it **generates** free text. For the three remaining tasks, it **scores** instead: the output space is enumerable, so a deterministic layer builds the candidates and the model only ranks them by the log-likelihood it assigns. It consults the model’s own scores, dictionary resources, and corpus statistics from the same enumerated corpora as the training data. Figure 3 shows why scoring is relatively compute-efficient: while free text generation needs one forward pass per output token, in scoring, a full candidate is scored in a single pass and whole batches of candidates are scored in parallel.

6.1 Machine Translation (Generation)

MT decodes greedily with $k = 3$ FMs retrieved from the full authentic+BT pool (the inference stage of Section 4).

After phase	MT	QA	SC	GC	MR	eqw
1 (step 24,000)	74.29	.576	.714	.630	.229	.578
2 (+3,500 cont.)	73.81	.570	.731	.746	.250	.607
3 (+ devel. fold)	hidden test: +.002 (Table 11)					

Table 7: Effect of each training phase on the development scores (eqw as in Table 6, best per column in bold). The phase-2 continuation converts GC headroom at little cost elsewhere. Phase 3 trains on the development sets, so its effect is measurable only on the hidden test.

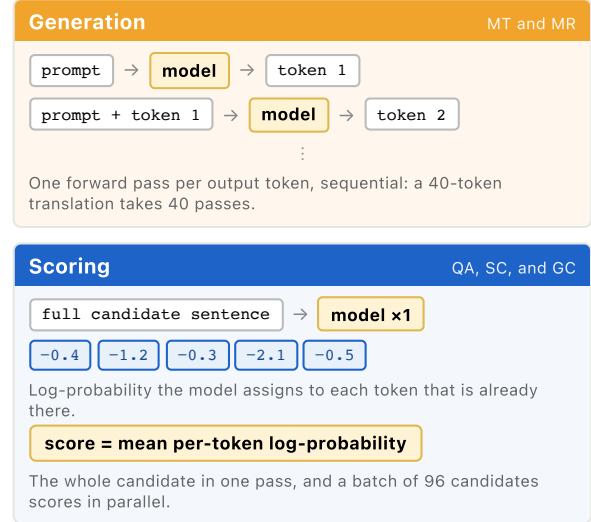


Figure 3: Generation and scoring use the same weights and the same forward computation. Generating runs one sequential forward pass per output token. Scoring runs one pass over a full candidate and reads off the log-probability of every token that is already there (values illustrative).

6.2 Question Answering (Scoring)

QA options are never free-generated. We place each option in the question context and score it by the log-likelihood the model assigns to it, following the standard likelihood-scoring convention for multiple choice (Brown et al., 2020). One question, for example, asks for the right reply to *Džakuju so!* (‘Thank you!’), offering exactly two options, *Tu prošu!* (‘Here you are!’) and *Hdže su moje knihi?* (‘Where are my books?’). Each option follows the answer prompt in turn, and in this example, the model’s log-likelihood prefers the first. Items where several questions share one option bank are solved jointly: one development cloze passage has twelve gaps drawing on a single 16-word option bank (four words are distractors), so scoring every word in every gap yields a 12×16 cost matrix, and an algorithm (Kuhn, 1955) fills all twelve gaps at once with the globally optimal one-to-one assignment, preventing the same option from being spent twice. The same joint setting appears in the

sentence-cloze task of Kong et al. (2020), which resolves it by exhaustively trying candidate assignments, where the assignment algorithm reaches the same optimum directly. Remaining items take the per-slot argmax.

6.3 Spell and Grammar Checking (Scoring)

Spell checking. SC must find the misspelt word in a sentence and produce its exact correct spelling. The model approaches this as a text generation task: prompted in the tagged format of Appendix C, it outputs the word it considers misspelt together with a proposed correction, and the flagged word is kept. Producing the exact corrected form is where a small model is weakest, so we additionally employ a dictionary layer to compensate. Drawing on the hunspell (Németh, 2026) lexicon for each language, soblex for Upper Sorbian (Baier et al., 2026) and dsb-spell for Lower Sorbian (derbnik, 2024), the engine retrieves candidate corrections for the flagged word: dictionary words within edit distance two of it (Burkhard and Keller, 1973), other inflected forms of the same word, and diacritic variants. Every candidate is substituted into the sentence, the resulting sentences are scored in one batch (Figure 3), and the best-scoring correction wins. This division of labour, where a dictionary proposes candidates and a language model picks among them in context, is the classical noisy-channel approach to spelling correction (Kernighan et al., 1990; Brill and Moore, 2000), applied to Slavic spelling and diacritics by Korektor (Richter et al., 2012). Here the fine-tuned multitask model itself plays the language-model role. One example illustrates how this works. In a sentence containing *twarški*, a word absent from the lexicon, the diacritic generator proposes *twarški* (‘construction’, adjectival), and the corrected sentence scores a log-likelihood of -3.84 against -4.39 for leaving the sentence unchanged, so the correction is applied. It matches the gold correction, and for this word the edit-distance and paradigm generators contribute no candidates.

Grammar checking. GC uses the same approach as the spell-checking task, but focuses on grammatical forms rather than spellings: the candidates are the inflected forms of the flagged word. Corrections the model flags itself always enter the output, and the engine may add one of its own only when two independent signals agree: its scoring margin is high, and the proposed form actually occurs next

to its neighbouring words in the reference corpora (a bigram witness). Prior work accepts a correction whenever a language model scores it above nearby variants, trusting the model’s preference alone (Yasunaga et al., 2021). The bigram witness adds a stricter, external requirement: the proposed form must also actually occur next to the same neighbouring words somewhere in the reference corpora. To our knowledge, requiring this corpus attestation on top of the model’s own preference is new. The exact decision rule and every fitted threshold are in Appendix B. For example, in *...dokument može so wot 1. awgust 2022 tež z pomocu widejoweje komunikacije wobkrućić* (‘...the document may be confirmed from 1 August 2022 also via video communication’, with *awgust* left in the nominative where the ordinal date requires the genitive), the model flags nothing, and the engine proposes *awgust* $\rightarrow$ *awgusta* from 302 paradigm candidates with a scoring margin of 0.059 and a witness value of 5.013. With the hsb weight of 0.1 the combined evidence reaches 0.561, above the hsb threshold of 0.42, so the correction is applied. *awgusta* is the gold form, so the corpus witness recovers a correct fix that the scoring margin alone would have left in place.

6.4 Maths Reasoning (Generation)

MR generates a worked solution and extracts the final numeric answer. Empty or unparseable answers enter a three-stage retry ladder: two strict retries, then one sampled retry at temperature 0.7, and a notation post-formatter normalises bare LaTeX in the answer field. The two strict retries target the two common failure modes, a truncated chain and a missing answer tag (the second appends an explicit answer-now continuation), and the sampled retry is a last resort. On the hidden test, 74.4% of items are answered on the first attempt, 7.2% recovered at the first retry and 18.4% at the second, and none required the sampled retry. For *Jedyn šat žada sej 2 kuli módreje wołmy a poł tak wjele běžeje wołmy. Kelko kulow trjebaš cyłkownje?* the model reasons in English (*The number of white yarn balls is $2/2=1$. So the total number of yarn balls is $2+1=3$.*) and emits `<answer> 3 </answer>`, the gold answer, without triggering the retry ladder.

6.5 Validating the Inference Strategies

Measured under the same development-set protocol as Figure 2, Table 8 isolates the inference strategies within the submission lineage. These infer-

Submission lineage	MT	QA	SC	GC	MR
plain decoding	72.04	.60	.71	.74	.17
full strategies (as deployed)	72.05	.78	.75	.77	.21

Table 8: Development-set scores of the submission-lineage weights with and without the inference strategies. The strategies drive QA, SC, GC, and MR, leaving MT nearly untouched.

ence strategies improve the performance of four of the five tasks, with MT remaining essentially unchanged (Table 8).

7 Submissions to the shared task

We submitted five systems in a $2 \times 2 + 1$ design (Table 9), **FullStack-DevTransfer** (the primary system), **FullStack-EarlyStop**, **FullStack-HeldOutDev**, **Plain-DevTransfer**, and **Plain-EarlyStop**, built so that pairwise comparisons on the hidden test each isolate one design decision: the inference strategies, the checkpoint-selection protocol, or the use of the development fold. The Plain systems run the same weights as their FullStack counterparts with the deterministic inference layer switched off, isolating the inference strategies of Section 6. With the layer off, every task is answered by free-text generation from the same base prompts (Appendix C): QA and the two checking tasks take the model’s generated string directly, with no candidate scoring or dictionary retrieval, and MR runs without the retry ladder. MT is unchanged, since its FM retrieval is part of the translation method rather than of the inference layer. The Plain systems are therefore pure text generation. Contrasting EarlyStop with DevTransfer isolates the checkpoint-selection protocol of Section 5. FullStack-HeldOutDev runs the phase-2 weights, which were never trained on the development sets and are therefore selected directly on them, isolating the development fold.

8 Results

Official results. Table 10 reports the organisers’ final evaluation. Our primary system, FullStack-DevTransfer, was designated joint winner of the Sorbian track. It wins QA (65.05, +7.72 accuracy points over the next-best system) and MR (30.40, +1.40), and places second on MT, SC, and GC. The two tasks on which our submission ranks first, QA and MR, are answered with the additional inference strategies of Section 6. These strategies are also applied to spelling and grammar checking, where our submission ranks second.

System	Selection	Devel.	Strategies
FullStack-DevTransfer [†]	DevTransfer	in	full
FullStack-EarlyStop	EarlyStop	in	full
FullStack-HeldOutDev	direct	out	full
Plain-DevTransfer	DevTransfer	in	off
Plain-EarlyStop	EarlyStop	in	off

Table 9: The five submitted systems ([†] = primary). Selection = checkpoint-selection protocol, where direct means selected on the development sets themselves. Devel. = whether the development fold is in the training mixture. Each Plain system shares its weights with the FullStack system above it, so Plain-vs-FullStack contrasts measure the inference strategies alone. We use these system names consistently throughout the paper.

Team	MT	QA	SC	GC	MR	Points
LT3 (ours)	69.57	65.05	76.11	78.32	30.40	22
HeyBusan	72.75	55.92	78.03	79.52	29.00	22
Hitsz	63.12	57.33	73.13	62.11	18.60	16
HSE	61.50	46.77	63.63	50.02	5.00	9
baseline	21.68	43.39	6.66	1.34	5.40	5

Table 10: Official Sorbian-track results (organisers’ final evaluation, metrics as in Table 1, best per task in bold). LT3 and HeyBusan were the joint winners of the track based on the total rank points summed over all tasks.

Test-scored ablations (proxy metrics). Table 11 compares the five submitted systems on the hidden test as scored by the submission-phase OCELOT leaderboard, whose **proxy metrics** differ from the official ones (SC and GC as joint both-field accuracy, stricter than the official mean of detection and correction, and MR without mathematical-equivalence checking), so these numbers are never comparable to Table 10. The inference strategies are the dominant factor: switching them off (FullStack-DevTransfer vs Plain-DevTransfer) costs 14.7 QA points, 10.1 SC, 4.6 GC, 6.8 MR, and 0.3 MT (Figure 4). This agrees with the development-set results in Table 8, where we measured the same strategies before the test phase.

Measured at a fixed submission-phase snapshot, before the final SC and MR levers landed, the strategies were worth +.049 on the leaderboard’s aggregate score (per-task QA +14.7, GC +4.6, SC +4.4, MR +0.4, MT +0.3). The other two components are small by comparison: checkpoint selection by transfer rather than cross-entropy early stopping is worth +.007 aggregate, and the development fold +.002. MT barely moves across all five systems (68.6 to 69.1), consistent with the strategies leaving generation untouched apart from exemplar retrieval.

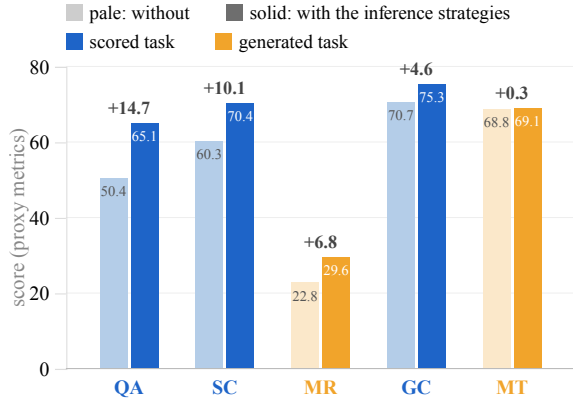


Figure 4: Hidden-test scores per task with and without the inference strategies (Plain-DevTransfer vs FullStack-DevTransfer, proxy metrics). The scored tasks (blue) gain the most, led by QA, while MT, where the strategies only add exemplar retrieval, barely moves. MR (generated) gains through the retry ladder and development-set retrieval guards.

System	MT	QA	SC	GC	MR
FullStack-DevTransfer	69.1	65.1	70.4	75.3	29.6
FullStack-HeldOutDev	69.1	63.6	64.6	75.0	24.0
FullStack-EarlyStop	69.0	63.9	64.5	74.5	21.8
Plain-DevTransfer	68.8	50.4	60.3	70.7	22.8
Plain-EarlyStop	68.6	49.4	60.9	68.9	22.2

Table 11: Hidden-test scores of the five submitted systems under the submission-phase proxy metrics (defined in the accompanying paragraph), not comparable to the official numbers in Table 10. Plain rows share weights with their FullStack counterparts (Table 9), so the FullStack-vs-Plain gaps measure the inference strategies alone.

9 Discussion and Conclusion

Two findings stand out. Firstly, the additional inference strategies help. Our submission ranks first on QA and MR and second on SC, GC and MT. Switching the strategies off costs 14.7 proxy points on QA and 10.1 on SC, while MT, where they only add exemplar retrieval, moves by 0.3 (Figure 4). The largest gains occur on the three scored tasks, which suggests that, for a 2B model, the bottleneck is producing exact surface forms. In other words, it is able to judge whether a word is (in)correct but is not able to give the corrected form without additional help. Letting the model choose between candidate answers, rather than write them, therefore plays to its strength. MR, which remains a generation task, gains instead from the retry ladder and the development-set retrieval.

Secondly, the MT results highlight the importance of both how the model is trained and how retrieval is used at training and inference. The

retrieval-based approach only pays off with sufficient training and when exemplars are retrieved from the authentic sentence pool during training. With limited training, fuzzy and random training exemplars perform similarly on the development sets. The difference emerges with longer training: only the fuzzy-on-authentic recipe keeps improving, yielding +3.63 chrF++ ahead at convergence, while every other recipe peaks early and degrades (Figure 2). A similar pattern emerges for BT placement: under FM retrieval, authentic-only fine-tuning yields +6.26 ahead of BT-containing rows. Both design choices from Section 4 therefore become visible only when training to convergence, highlighting a potential limitation of exemplar-selection studies conducted under small, fixed training budgets. Importantly, this preference for authentic data applies to training, not inference. Across the converged optima, enlarging the inference-time retrieval pool with BT pairs never hurts under FM retrieval and yields modest gains of up to +0.2 chrF++, whereas random exemplar selection cannot exploit the larger pool.

To conclude, we fine-tuned a single 2B model to serve five tasks in Upper and Lower Sorbian. We first focused on the translation task: a strong MT model let us back-translate the large monolingual corpora into synthetic parallel data, and that data enlarged the fuzzy-match pool the final model retrieves from at inference. We then folded the remaining tasks into the same weights and answered them by scoring candidates derived from dictionaries and corpora. This combination of retrieval and scoring made our system joint winner of the Sorbian track. All artefacts, code, weights, and data are publicly available (Appendix D).

Limitations

The main limitation is scope in the sense that everything here is measured on one track, two closely related West Slavic languages, and one 2B base model. We make no claim that the recipe transfers to other language pairs or model scales. More generally, we designed the submission to maximise the shared task’s automated metrics. For tasks where human judgement is the real standard, such as MT, a human evaluation could tell a different story than the automated metrics.

Further, we chose several inference thresholds to maximise development-set scores, so some measured gains may not carry over to unseen data, and

the reported gains from these fitted components should be read as upper estimates.

Finally, all cross-system ablations (Table 11) use the submission-phase proxy metrics, which differ from the official metrics for SC, GC, and MR (Section 8), so the two result families are not directly comparable.

Acknowledgements

We thank the WITAJ-Sprachzentrum for the Sorbian data underlying the task, in particular Beate Brezan, Anita Hendrichowa, and Marko Meskank, with maths annotation by Krystof Persin and language certificate materials by Tomas Solta. We thank the organisers at TUM for running the shared task and for their responsiveness throughout. The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, FWO and the Flemish Government.

References

- Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. 2023. [In-context examples selection for machine translation](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8857–8873.
- Duarte M. Alves, Nuno M. Guerreiro, João Alves, José Pombal, Ricardo Rei, José G. C. de Souza, Pierre Colombo, and André F. T. Martins. 2023. [Steering large language models for machine translation with finetuning and in-context learning](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11127–11148, Singapore. Association for Computational Linguistics.
- Bernhard Baier, Wito Bejmak, and Serbski institut. 2026. soblex: Upper Sorbian spelling dictionary and hunspell spellchecker. <https://soblex.de>. Version 3.09.18.
- Eric Brill and Robert C. Moore. 2000. [An improved error model for noisy channel spelling correction](#). In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 286–293.
- Tom Brown, Benjamin Mann, Nick Ryder, et al. 2020. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems* 33.
- Bram Bulté and Arda Tezcan. 2019. Neural fuzzy repair: Integrating fuzzy matches into neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1800–1809.
- W. A. Burkhard and R. M. Keller. 1973. Some approaches to best-match file searching. *Communications of the ACM*, 16(4):230–236.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. 2021. [Training verifiers to solve math word problems](#). Preprint, arXiv:2110.14168.
- derbnik. 2024. dsb-spell: Lower Sorbian spellchecking dictionary. <https://dsb-spell.sourceforge.net>. Version 1.5.0.7.
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems* 36.
- Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. [Understanding back-translation at scale](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500.
- Alexander Fraser. 2020. [Findings of the WMT 2020 shared tasks in unsupervised MT and very low resource supervised MT](#). In *Proceedings of the Fifth Conference on Machine Translation*, pages 765–771.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the MATH dataset](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.
- Cuong Hoang, Devendra Sachan, Prashant Mathur, Brian Thompson, and Marcello Federico. 2023. [Improving retrieval augmented neural machine translation by controlling source and fuzzy-match interactions](#). In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 289–295.
- Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. [Surface form competition: Why the highest probability answer isn’t always right](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.
- Berivan Isik, Natalia Ponomareva, Hussein Hazimeh, Dimitris Paparas, Sergei Vassilvitskii, and Sanmi Koyejo. 2025. [Scaling laws for downstream task performance in machine translation](#). In *The Thirteenth International Conference on Learning Representations*.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.
- Mark D. Kernighan, Kenneth W. Church, and William A. Gale. 1990. [A spelling correction program based on a noisy channel model](#). In *COLING 1990 Volume 2: Papers Presented to the 13th International Conference on Computational Linguistics*.

- Woosung Koh, Jeyoung Jeon, Youngjin Song, Yujin Cheon, Soowon Oh, Jaehyeong Choi, and Se-Young Yun. 2026. [mSFT: Addressing dataset mixtures overfitting heterogeneously in multi-task SFT](#). *Preprint*, arXiv:2603.21606.
- Xiang Kong, Varun Gangal, and Eduard Hovy. 2020. [SCDE: Sentence cloze dataset with high quality distractors from examinations](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5668–5683.
- Harold W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97.
- Samuel Larkin, Chi-kiu Lo, and Rebecca Knowles. 2025. [NRC systems for the WMT2025-LRSL shared task](#). In *Proceedings of the Tenth Conference on Machine Translation*, pages 1133–1142.
- Jindřich Libovický and Alexander Fraser. 2021. [Findings of the WMT 2021 shared tasks in unsupervised MT and very low resource supervised MT](#). In *Proceedings of the Sixth Conference on Machine Translation*, pages 726–732.
- Thomas Moerman, Els Lefever, and Arda Tezcan. 2026. Fuzzy match augmentation with back-translation for domain-specific machine translation: An empirical study on large language models and neural machine translation. Under review.
- Yasmin Moslem, Rejwanul Haque, John D. Kelleher, and Andy Way. 2023. [Adaptive machine translation with large language models](#). In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 227–237.
- 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.
- László Németh. 2026. Hunspell: spell checker and morphological analyser. <https://hunspell.github.io>.
- Shu Okabe, Daryna Dementieva, Marion Di Marco, Lukas Edman, Katharina Haemmerl, Marko Měškank, Anita Hendrichowa, and Alexander Fraser. 2025. [Findings of the WMT 2025 shared task LLMs with limited resources for Slavic languages: MT and QA](#). In *Proceedings of the Tenth Conference on Machine Translation*, pages 503–519, Suzhou, China. Association for Computational Linguistics.
- Shu Okabe, Marion Di Marco, Kathy Hämmerl, Daryna Dementieva, Lukas Edman, Marko Měškank, Anita Hendrichowa, and Alexander Fraser. 2026. Findings of the WMT26 Shared Task: Multitask LLMs with Limited Resources. In *Proceedings of the Eleventh Conference on Machine Translation*, Budapest, Hungary. Association for Computational Linguistics.
- Taido Purason and Mark Fishel. 2025. [TartuNLP at WMT25 LLMs with limited resources for Slavic languages shared task](#). In *Proceedings of the Tenth Conference on Machine Translation*, pages 1143–1150.
- Qwen Team. 2026. [Qwen3.5: Towards native multimodal agents](#). Qwen Team blog.
- Michal Richter, Pavel Straňák, and Alexandr Rosen. 2012. [Korektor – a system for contextual spell-checking and diacritics completion](#). In *Proceedings of COLING 2012: Posters*, pages 1019–1028.
- Hossain Shaikh Saadi, Minh Duc Bui, Mario Sanz-Guerrero, and Katharina von der Wense. 2025. [JGU Mainz’s submission to the WMT25 shared task on LLMs with limited resources for Slavic languages](#). *Preprint*, arXiv:2509.22490.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. [Improving neural machine translation models with monolingual data](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.
- Arda Tezcan, Alina Skidanova, and Thomas Moerman. 2024. [Improving fuzzy match augmented neural machine translation in specialised domains through synthetic data](#). *The Prague Bulletin of Mathematical Linguistics*, 122:9–42.
- David Vilar, Markus Freitag, Colin Cherry, Jiaming Luo, Viresh Ratnakar, and George Foster. 2023. [Prompting PaLM for translation: Assessing strategies and performance](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15406–15427.
- Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems* 33.
- Marion Weller-Di Marco and Alexander Fraser. 2022. [Findings of the WMT 2022 shared tasks in unsupervised MT and very low resource supervised MT](#). In *Proceedings of the Seventh Conference on Machine Translation*, pages 801–805.
- Jitao Xu, Josep Crego, and Jean Senellart. 2020. [Boosting neural machine translation with similar translations](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1580–1590.
- Michihiro Yasunaga, Jure Leskovec, and Percy Liang. 2021. [LM-Critic: Language models for unsupervised grammatical error correction](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.
- Kyra Yee, Yann Dauphin, and Michael Auli. 2019. [Simple and effective noisy channel modeling for neural machine translation](#). In *Proceedings of the*

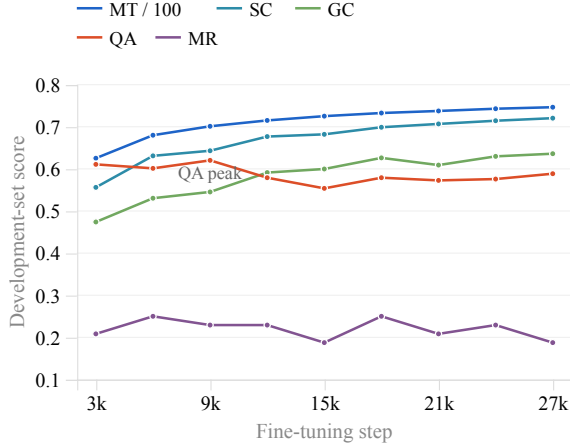


Figure 5: Five-task development scores along the phase-1 trajectory (plain decoding, MT plotted /100). MT, SC, and GC rise for the whole run, QA peaks at 9k and erodes, and MR is noisy-flat.

2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 5696–5701.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. [Calibrate before use: Improving few-shot performance of language models](#). In *Proceedings of the 38th International Conference on Machine Learning*.

A Training-Length Trade-off

Figure 5 plots the full trajectory behind the cross-task trade-off of Section 5 (Table 6). The run uses the training mixture of Section 5 and the hyperparameters of Table 12 on an annealed schedule truncated at 27k of 36k planned steps, with checkpoints every 3,000 steps, each evaluated on the gold development sets with plain decoding and MT $k = 3$ fuzzy retrieval. Full per-checkpoint numbers accompany the released artefacts (Appendix D).

B Hyperparameters and Final Configurations

Table 12 lists every fitted inference value of the primary system together with the shared training hyperparameters. The GC engine may act alone (union arbitration) only when $\text{margin} + w \cdot \text{witness} \geq t$ with the per-language (w, t) below. The *margin* is the null-margin of the scoring engine: the model’s log-likelihood of the sentence with the best candidate correction substituted minus that of the unedited sentence. The *witness* is a corpus bigram-attestation score, $\text{witness} = \text{bg}(\text{candidate}) - \text{bg}(\text{observed})$, with $\text{bg}(w) = \log(c_{\text{left},w} + 0.5) + \log(c_{w,\text{right}} + 0.5)$, where c

Component	Parameter	Value
MT	decoding	greedy
MT	exemplars, training	uniform sampling (0/1), 50% carriers, authentic, cap 100k
MT	exemplars, inference	$k = 3$ (authentic+BT)
QA	option scoring	in-place log-likelihood
QA	shared option banks	Hungarian assignment
SC/GC	candidate engine	BK-tree edit distance 2
SC/GC	scoring batch / cands	96 / 24
GC	paradigm cap	60
GC	witness hsb (w, t)	(0.1, 0.42)
GC	witness dsb (w, t)	(0.4, 0.76)
SC	paradigm cap	30
SC	diacritic top-up	dsb only, min freq 5
SC	non-word top-up τ	hsb 0.30, dsb 0.36
MR	retry ladder	2 strict + 1 sampled ($T=0.7$)
MR	development retrieval	similarity ≥ 0.95
all	LoRA rank / α	256 / 512
all	learning rate	2e-5 (phase 1) 1e-5 (continuations)
all	batch / grad. accum.	8 / 2
all	sequence length	1024
all	LR schedule	linear anneal to 0
all	generation seed	1234

Table 12: Fitted inference configuration of the primary system FullStack-DevTransfer. SC/GC engine rows are shared. The two MT exemplar rows are the two retrieval stages of Section 4.

counts the bigram of the word w with its left or right neighbour in the reference-corpus bigram table. We fitted the witness weight and threshold via an id-parity cross-fit on the gold development sets.

C Prompt Formats

All prompts are rendered through the model’s chat template as single user messages, with thinking disabled. The templates below are verbatim from the released inference code.

MT. Exemplars precede the source as plain language-labelled lines. The inference prompt carries $k = 3$ exemplar pairs, and a training carrier prompt carries exactly one (Section 4). Zero-shot training rows omit the exemplar lines entirely.

```
Translate the source text from
{src_lang} to {tgt_lang}
```

```
{src_lang}: {exemplar source}
{tgt_lang}: {exemplar target}
...
{src_lang}: {source}
{tgt_lang}:
```

SC and GC. Both use a tagged two-field response format, differing only in the error type named in the instruction (misspelled word for SC, grammatically incorrect word for GC):

```
The following sentence contains at most
one misspelled word. Respond with
```

exactly these two tags and nothing else:
`<wrong> misspelled word </wrong>`
`<corrected> correct spelling </corrected>`
 If the sentence has no error, use the
 literal string CORRECT in both tags:
`<wrong> CORRECT </wrong>`
`<corrected> CORRECT </corrected>`
`<sentence> {sentence} </sentence>`

QA. Used for scoring rather than free generation (Section 6): each option is placed after `Answer:` and scored in place.

Answer the multiple-choice question with the number of the correct option only.

```
{context}
Question: {question}
{n}. {option n}
...
Answer:
```

MR.

Answer this mathematical reasoning question. Place your final answer, using LaTeX when appropriate, in this format:
`<answer> answer </answer> {question}`

An example. The MT template rendered on the development source of Figure 1 with its actually retrieved exemplars (line breaks added for layout). The training render shows the carrier case, whose single exemplar is an authentic pair. The inference render carries three FMs, here all back-translated pairs.

Translate the source text from German to Upper Sorbian

German: Die Kinder und die Eltern haben das Sprachangebot gut angenommen.
 Upper Sorbian: Džěći a starši su rěčny poskitk derje přiwzali.
 German: So erleben die Kinder Freude mit und an der Sprache.
 Upper Sorbian:

Translate the source text from German to Upper Sorbian

German: Sie lernen aber auch mit der Geschichte der Kunst.
 Upper Sorbian: Zeznajomjeja pak so tež ze stawiznami wuměłstwa.
 German: So konnten sie sich besser auf das Erlebnis und die Gemeinschaft konzentrieren.
 Upper Sorbian: Tak móžachu so lěpje na dožiwjenje a na zhromadnosć koncentrować.
 German: Ihre Freude an der Gesang und dem Spielen wird schnell auf das Publikum übertragen.
 Upper Sorbian: Jich radosć ze spěwanja kaž tež hudženja so spěšnje na publikum přenese.
 German: So erleben die Kinder Freude mit und an der Sprache.
 Upper Sorbian:

D Reproduction Details

All artefacts are public: the code repository with the primary system’s configuration, an end-to-end reproduction script, and a retraining script,⁴ the

⁴<https://github.com/TomMoeras/wmt26-lt3-sorbian>

merged weights,⁵ and a data repository with the retrieval pools and training sets.⁶

The computational cost is modest: reproducing the primary system end-to-end from the public artefacts, from the fine-tuned weights to all five task outputs, takes about 2 hours on a single H200 GPU. Training and development ran on the VSC infrastructure acknowledged above.

⁵<https://huggingface.co/TomMoeras/wmt26-lt3-sorbian>

⁶<https://huggingface.co/datasets/TomMoeras/wmt26-lt3-sorbian-data>