CUNI and Phrase at WMT25 MT Evaluation Task

Miroslav Hrabal,[1] Ondřej Glembek,[2] Aleš Tamchyna,[2] A. Silja Hildebrand,[2] Alan Eckhardt,[2] Miroslav Štola,[2] Sergio Penkale,[2] Zuzana Šimečková,[2] Ondřej Bojar,[1] Alon Lavie,[2] Craig Stewart[2]

[1]Charles University, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
{hrabal,bojar}@ufal.mff.cuni.cz

[2]Phrase a.s.
name.surname@phrase.com

Abstract

This paper describes the joint effort of Phrase a.s. and Charles University's Institute of Formal and Applied Linguistics (CUNI/UFAL) on the WMT25 Automated Translation Quality Evaluation Systems Shared Task. Both teams participated both in a collaborative and competitive manner, i.e. they each submitted a system of their own as well as a contrastive joint system ensemble. In Task 1, we show that such an ensembling—if chosen in a clever way—can lead to a performance boost. We present the analysis of various kinds of systems comprising both "traditional" NN-based approach, as well as different flavours of LLMs—off-the-shelf commercial models, their fine-tuned versions, but also in-house, custom-trained alternative models. In Tasks 2 and 3 we show Phrase's approach to tackling the tasks via various GPT models: Error Span Annotation via the complete MQM solution using non-reasoning models (including fine-tuned versions) in Task 2, and using reasoning models in Task 3.

1 Introduction

Machine translation (MT) evaluation has evolved rapidly in recent years, driven by the dual rise of large language models (LLMs) and specialised neural quality estimation (QE) architectures. Shared tasks, such as the WMT MT Evaluation campaign, provide a rigorous and reproducible framework for assessing advances in automated evaluation. The WMT25 MT Evaluation Task continues this tradition, offering a benchmark for comparing diverse approaches across multiple subtasks, from system-level quality prediction to fine-grained error annotation.

In this paper, we present a joint study by Phrase a.s. and Charles University's Institute of Formal and Applied Linguistics (CUNI/UFAL), combining independent research streams with collaborative experimentation. Both teams participated in Task 1, each submitting their own primary and secondary systems, as well as a contrastive joint ensemble in Task 1. This setting allowed us to examine not only the relative strengths of distinct modelling paradigms but also the potential synergies of cross-team integration. Phrase also participated in Task 2 and Task 3.

Our contributions are threefold. First, we investigate complementary modelling strategies for Task 1, including NN-based regression models trained on large-scale MQM-style data (multidimensional quality metrics, Lommel et al., 2013), and LLM-based systems producing full linguistic quality annotations. We analyse how fusing/ensembling these approaches, when carefully configured, can yield measurable performance gains. Second, for Task 2, we evaluate GPT-based AutoLQA (Automatic Language Quality Assessment) systems—both off-the-shelf and fine-tuned variants—demonstrating their applicability to error span annotation in MQM and ESA (error span annotation, Kocmi et al., 2024) formats. Third, for Task 3, we explore reasoning-enabled LLMs with targeted prompting for translation improvement, highlighting trade-offs between fluency- and accuracy-oriented objectives.

By documenting both competitive and collaborative results, we aim to provide insight into practical design choices—such as metric selection, data sampling, and system fusion—that influence performance in automated MT evaluation. Beyond the leaderboard standings, our analysis identifies patterns that may inform future research on hybrid evaluation architectures, the role of fine-tuning in LLM-based evaluators, and the limits of current metrics in capturing subtle quality differences.

Proceedings of the Tenth Workshop on Machine Translation (WMT), pages 934–944

2 Task 1

In Task 1, Phrase and UFAL decided to collaborate and choose the strategy in which each organisation submits its own primary system and one of its own secondary systems. Each team dedicated the other secondary slot to a joint system ensemble. The submitted systems are marked with an asterisk* in the respective tables.

2.1 Phrase Systems

We have two sets of models that we experimented with in Task 1: (1) NN-based quality regression models that directly output the quality prediction score, and (2) LLM-based models that run the full linguistic quality assessment that is used to calculate the final score, as described below.

There are two important decisions that we made in the course of the evaluations mainly due to our internal research conventions and momentum, and also due to some technical difficulties.

The first decision was about the development metric for Task 1. Although acc^*_{eq} *tie-calibrated pairwise accuracy* (Deutsch et al., 2023) is the primary metric of the WMT25 evaluations, we used Kendall's τ and Pearson r coefficients as our development proxy metrics. The main reason for this decision was our late adoption and implementation issues of the acc^*_{eq} metric. Note that we also used the two proxy metrics when fusing the CUNI and the Phrase systems.

The second decision was about the submission score. We decided to treat all scores in a unified MQM/ESA fashion, i.e. internally we do not distinguish between the two scores. The first reason for this adoption is that the evaluation is insensitive to the dynamic range of the two metrics and—to our knowledge—the difference between the two is (1) lower penalization of the "Fluency/Punctuation" MQM category and (2) higher penalization of the non-translations. The second reason is rather practical: at Phrase, our tools (that we used off-the-shelf for this evaluation) internally work with a single score (let us refer to the score merely as MQM) that is defined as follows: Given the input segment $\mathcal{X}$ (with its source and hypothesis fields), we compute the MQM score as:

$$\mathrm{MQM}(\mathcal{X}) = \max\left\{0, 1 - \frac{\mathrm{pen}\,(\mathcal{X})}{|\mathcal{X}_{\mathrm{source}}|}\right\}, \quad (1)$$

where $|\cdot|$ denotes the word count (note that we compute the word count of the source segment), and $\mathrm{pen}(\cdot)$ is the sum of all penalty points retrieved from the AutoLQA system output, where we map the minor, major and critical errors to the penalties of 1, 5 and 5, respectively (i.e., no quantitative difference between major and critical errors).

2.1.1 Phrase Development Data

As development data we used the Google-MQM (Freitag et al., 2021) dataset and the WMT22-QE combined dev and test sets. Since the latter only covers three language pairs (`en-de`, `en-ru` and `zh-en`, 1500 segments each), we also pulled test data from the Google-MQM set, which covers seven language pairs (`en-de`, `en-es`, `en-ru`, `en-zh`, `he-en`, `ja-zh`, `zh-en`). We randomly selected 700 segments per LP while making sure that only one system's translation of a source sentence was included. We then removed all translation variants of those source sentences from the remaining data to prevent contamination and filtered out very self-similar segments as well, resulting in a training dataset of about 150k segments.

2.1.2 Phrase QPS Systems

One set of features for the ensembles is provided by research variants of the Phrase QPS (Quality Performance Score) system[1][2] (*pqps*). We experimented with three pre-trained models: XLMR-large (Conneau et al., 2019), GTE (Zhang et al., 2024), and RemBERT (Chung et al., 2021), which we selected not only according to their accuracy, but also with regard to memory and inference speed requirements for the production environment at Phrase.

We continued training on a large amount of internal Phrase post-edit data covering 226 language pairs with chrF (Popović, 2015) as gold-label (resulting models are tagged *pe* in the result tables.)

Then we further fine-tuned on combined data from internal Phrase MQM-annotated and post-edit data as well as our training split of the Google-MQM and Amazon-bio-MQM datasets (Zouhar et al., 2024). We experimented with three gold/reference score variants: a balanced mix of MQM-score and chrF labels (*bal*), the same mix with a smoothed version of the MQM score (*balsm*) and one variant with only MQM- and MQM-like-scores (*mqm*) as gold label. In all cases, Equa-

[1] https://phrase.com/phrase-quality-technologies/quality-performance-score/

[2] https://support.phrase.com/hc/en-us/articles/5709672289180-Phrase-QPS-Overview

tion (1) was the fundamental instrument for the MQM gold label score calculation.

See Table 1 for individual system performance on our development test sets (see Section 2.1.1). Due to the limited number of slots, we did not submit the best Phrase QPS system.

2.1.3 Phrase AutoLQA System

The AutoLQA system (Automatic Linguistic Quality Assessment) is an LLM-based system whose underlying model is a fine-tuned `gpt-4o-mini` model, marked as `gpt-4o-mini-FT`. This is the very same system that was used for the primary submission of Task 2 (see Section 3 for reference). The system produces complete MQM annotations based on which the MQM-score is produced via Equation (1).

This system is referred to as *palqa-wmt25* in Table 1.

2.2 CUNI/UFAL Systems

This section describes the systems submitted by the CUNI team.

2.2.1 Training and Development data

For training, we used a subsample of WMT24 ESA scores, counting only 2160 examples in total.

Because high ESA scores are much more frequent in the full dataset, we resampled the data to produce a more uniform score distribution, while ensuring an even spread across language pairs.

As the development set, we used WMT23 metrics task en2cs DA scores. Due to the computational cost of the evaluation, we did not use other language pairs for the evaluation during development.

2.2.2 Systems based on Gemma 3 27B-it and DSPy

Our primary submission (*mr7.2.1*, in Table 2) and one of the secondary submissions (*mr6*, in Table 1 and Table 2) were based on the Gemma 3 27B-it[3] used via the DSPy framework (Khattab et al., 2024). Its MIPROv2 optimiser (Opsahl-Ong et al., 2024) was used to automatically select n-shot examples and adjust the prompts for improved performance.

[3]For both optimisation and inference, we run several instances of the model using vLLM 0.9.2+rocm641, each instance on 2x AMD MI210 with 16k tokens context. We used a LiteLLM proxy server for load balancing between instances. We also used GNU Parallel (Tange, 2025) as a workaround for performance issues when trying to scale the number of concurrent requests to fully utilise GPUs.

We selected the Gemma 3 27B-it model for its relatively compact size and strong multilingual capabilities. Our focus was on using reasonably sized open-weight models that we can run on our hardware, so we chose not to use larger commercial models available through APIs, even though we expect them to perform better.

The optimisation target was essentially rescaled Mean Squared Error:

$$\text{score}(X) = \frac{1}{n}\sum_{i=1}^{n} 1 - \frac{(X_{i,\text{gold}} - X_{i,\text{pred}})^2}{100^2} \quad (2)$$

Both submissions first predict,in one or multiple LLM calls, seven integer scores (0–10) covering different translation quality dimensions:

- accuracy and completeness
- terminology and consistency
- fluency and coherence
- style tone and audience fit
- locale conventions and formatting
- technical integrity
- cultural appropriateness

These categories were inspired by MQM, with initial detailed descriptions drafted by GPT o3-mini, see Appendix B for details on the optimized prompts:

After predicting these dimensions, both *mr6* and *mr7.2.1* output an overall translation quality score (0–100 integer). Similarly to the approach taken by the Phrase team, we chose not to distinguish between pairs that solicit ESA scores and MQM scores and use this 0-100 score for both.

The key difference between the systems lies in the overall score aggregation:

- **mr6:** Predicts each dimension separately in isolation, with a separate request for each. The overall score is then predicted in a separate LLM call, using only the dimension scores (not the original text). This approach was motivated by the lack of gold data for dimension-level scores and the absence of a clear formula to combine them. We originally planned to replace this aggregation with a simpler regression model (e.g., linear regression or gradient

Table 1: Task 1 Individual models' performance. The asterisk * denotes submitted systems: *palqa-wmt25* is Phrase's secondary submission and *cuni-mr6-overall* is CUNI's secondary submission. Best scores in each group in bold.

		google-mqm		wmt22-qe	
Feature/Model	Type	Kendall's τ	Pearson r	Kendall's τ	Pearson r
pqps-gte-pe	NN	0.0776	0.1093	0.2185	0.3710
pqps-rembert-pe	NN	0.0970	0.1486	0.2996	0.4576
pqps-xlmr-pe	NN	0.0954	0.1401	0.2760	0.4298
pqps-gte-bal-v1	NN	0.2306	0.4538	0.2670	0.4293
pqps-gte-bal-v2	NN	0.2324	0.4547	0.2589	0.4349
pqps-rembert-bal	NN	0.2465	0.4855	0.3184	0.4830
pqps-xlmr-bal	NN	0.2568	0.4936	0.3083	0.4778
pqps-gte-balsm	NN	0.2295	0.3973	0.2554	0.4390
pqps-rembert-balsm	NN	0.2422	0.4106	**0.3235**	**0.4945**
pqps-xlmr-balsm-v1	NN	0.2164	0.4288	0.2491	0.4416
pqps-xlmr-balsm-v2	NN	0.2292	0.3736	0.2993	0.4804
pqps-gte-mqm	NN	0.2398	0.4660	0.0789	0.2828
pqps-rembert-mqm	NN	0.2406	0.4652	0.2572	0.4119
pqps-xlmr-mqm	NN	**0.2620**	**0.5110**	0.0058	0.0954
palqa-wmt25*	LLM	**0.2790**	**0.4987**	**0.3145**	**0.4017**
cuni-mr6-y0	LLM	0.1767	0.3058	0.2153	0.2308
cuni-mr6-y1	LLM	0.1794	0.3311	0.2176	**0.2851**
cuni-mr6-y2	LLM	0.1800	0.2711	0.2240	0.2008
cuni-mr6-y3	LLM	0.1923	0.3098	0.2139	0.2192
cuni-mr6-y4	LLM	0.1249	0.2085	**0.2332**	0.2297
cuni-mr6-y5	LLM	0.1647	0.2759	0.1886	0.1978
cuni-mr6-y6	LLM	**0.2026**	0.3386	0.2034	0.2259
cuni-mr6-overall*	LLM	0.1826	**0.3460**	0.2214	0.2481

boosting) after the initial MIPROv2 optimisation, but our preliminary experiments showed no consistent improvements. We therefore submitted the LLM-based aggregation version, although the lack of gain might be due to unidentified implementation issues.

- **mr7.2.1:** Predicts all dimensions and the overall score in a single LLM call.

Both systems use the chain-of-thought technique (Wei et al., 2022) as implemented by the DSPy.[4] If the model fails to produce the response in the correct format, there are 2 retries with different temperatures: 0.5 and $\frac{2}{3}$.

In addition, *mr7.2.1* includes a fallback mode that bypasses chain-of-thought and generates the final answer directly if reasoning fails in all 3 tries (most commonly due to getting stuck in generating repetitive loops).

The complete training and inference code for the models submitted by CUNI will be available on GitHub.[5]

We evaluated our systems on WMT23 metrics using the mt-metrics-eval tool. We show the results in Table 2.

To get some idea whether the MIPROv2 optimisation and chain-of-thought actually contributes to better performance, we evaluate several modifications of the *mr7.2.1* submission: *_noopt* is a version with no MIPROv2 optimisation and consequently no n-shot examples. *_nocot* is a version where the output is generated directly with no chain-of-thought reasoning. The *_nocot_noopt* variant disables both. We can notice that disabling either of these or both at the same time seems to worsen the performance in all of the tracked metrics.

[4] https://dspy.ai/api/modules/ChainOfThought/

[5] https://github.com/hrabalm/wmt25-mt-eval-task

Table 2: CUNI models' performance on WMT23 en2cs dev set: segment-level Pearson r, system-level Pearson r and segment-level accuracy with optimised tie threshold. Our systems are in bold, systems marked with ** are CUNI primary submissions and systems marked with * are CUNI secondary submissions.

Model	Seg. P r	Rank	Sys. P r	Sys. rank	acc^*_{eq}	acc^*_{eq} rank
XCOMET-Ensemble	0.4017	1	0.9025	1	0.5404	2
XCOMET-QE-Ensemble[noref]	0.3952	2	0.9082	1	0.5286	4
COMET	0.3771	3	0.8646	2	0.5239	5
BLEURT-20	0.3730	3	0.7935	3	0.5111	7
MetricX-23	0.3606	4	0.8914	1	0.5500	1
KG-BERTScore[noref]	0.3503	4	0.7899	3	0.5062	10
CometKiwi[noref]	0.3503	5	0.7898	3	0.5171	6
MetricX-23-QE[noref]	0.3477	5	0.8782	2	0.5392	3
cometoid22-wmt22[noref]	0.3411	6	0.8254	3	0.5012	11
mr7.2.1[noref]**	0.3320	6	0.8021	3	0.3858	25
mr7.2.1_noopt[noref]	0.3146	7	0.7452	4	0.3351	27
mr7.2.1_nocot[noref]	0.3144	7	0.7508	4	0.3483	26
mr6[noref]*	0.3142	7	0.8114	3	0.4078	24
mr7.2.1_nocot_noopt[noref]	0.3115	7	0.7494	4	0.3295	29
GEMBA-MQM[noref]	0.3094	7	0.8520	2	0.3295	28
MS-COMET-QE-22[noref]	0.2864	8	0.7965	3	0.4984	12
prismRef	0.2649	9	0.5571	5	0.4910	14
XLsim	0.2589	9	0.6268	4	0.5075	9
YiSi-1	0.2453	10	0.5677	4	0.4968	13
BERTscore	0.2283	11	0.4798	5	0.4899	15
tokengram_F	0.2031	12	0.4087	7	0.4817	17
chrF	0.2006	13	0.4495	6	0.4794	18
f200spBLEU	0.1986	13	0.4962	5	0.4793	19
BLEU	0.1857	14	0.5186	5	0.4612	23
embed_llama	0.1720	15	0.4661	6	0.4767	20
prismSrc[noref]	0.1710	15	-0.0416	7	0.4615	22
eBLEU	0.1689	15	0.4672	6	0.4837	16
mre-score-labse-regular	0.1298	16	0.7184	4	0.5089	8
Random-sysname[noref]	0.0018	17	0.0145	7	0.4646	21

2.3 Fusion / Ensembling

We used the features/models described above to train six standard regression models: Linear Regression (Freedman, 2005), Ridge Regression (Hoerl and Kennard, 1970), Decision Tree Regression (Breiman et al., 1984), Random Forest Regression (Breiman, 2001), Gradient Boosting Regression (Friedman, 2000), and MLP Regression (Rosenblatt, 1958), all with default scikit-learn (Pedregosa et al., 2011) hyper-parameters. The models were trained on our training data split of the Google-MQM public dataset (see Section 2.1.1). For both the Phrase-only ensembles and the Collaboration ensembles we compared using all features as well as a "slick" version which uses only the best feature of each type: *pqps-xlmr-bal*, *palqa-wmt25* and *cuni-mr6-overall*. See results in Table 3. The submitted systems are marked with an asterisk. We selected the Gradient Boosting Regressor because it seems to achieve a good balance across both testsets as well as both metrics.

For a language-pair specific breakdown of results for the submitted systems refer to Table 7.

3 Task 2 (Phrase)

In Task 2, we experimented with using our internal AutoLQA systems that are based on LLMs and in-context learning. AutoLQA systems have been developed to produce the complete MQM annotation, i.e. they include the error category by default.

Table 3: Task 1 Phrase-only and Collaboration ensembles for different regression models and feature sets. Note on the submitted systems: phrase-slick is Phrase's primary submission, collab-slick is Phrase's secondary submission, collab-full is CUNI's secondary submission

Feature Set	Model	google-mqm		wmt22-qe	
		Kendall's τ	Pearson r	Kendall's τ	Pearson r
phrase-full	Linear Regression	0.2829	0.5517	0.3133	0.5092
15 features	Ridge Regression	0.2829	0.5517	0.3137	0.5095
	Decision Tree Regressor	0.2720	0.4976	**0.3533**	0.5047
	Random Forest Regressor	**0.3292**	0.5455	0.3006	0.4936
	Gradient Boosting Regressor	0.2912	0.5531	0.3423	**0.5250**
	MLP Regressor	0.2970	**0.5641**	0.3214	0.5115
phrase-slick	Linear Regression	0.2902	**0.5562**	0.3423	**0.5153**
2 features	Ridge Regression	0.2902	**0.5562**	0.3423	**0.5153**
	Decision Tree Regressor	**0.2962**	0.5431	0.3451	0.5052
	Random Forest Regressor	0.2706	0.5175	0.2937	0.4459
	Gradient Boosting Regressor**	0.2931	0.5549	**0.3474**	0.5145
	MLP Regressor	0.2921	0.5543	0.3444	0.5118
collab-full	Linear Regression	0.2844	0.5529	0.3179	0.5113
23 features	Ridge Regression	0.2842	0.5526	0.3184	0.5115
	Decision Tree Regressor	0.2720	0.4976	**0.3533**	0.5047
	Random Forest Regressor	**0.3327**	0.5553	0.2990	0.4940
	Gradient Boosting Regressor*	0.2931	**0.5558**	0.3455	**0.5302**
	MLP Regressor	0.2732	0.5369	0.3249	0.5145
collab-slick	Linear Regression	0.2934	0.5580	0.3424	0.5178
3 features	Ridge Regression	0.2934	**0.5581**	0.3424	0.5178
	Decision Tree Regressor	0.2930	0.5397	0.3507	0.5099
	Random Forest Regressor	0.2824	0.5209	0.3096	0.4660
	Gradient Boosting Regressor*	**0.2939**	0.5524	**0.3558**	**0.5213**
	MLP Regressor	0.2812	0.5484	0.3365	0.5108

Table 4: Task 2 F1 score for various engines. The results are reported on our Google-MQM test data. Note the `gpt-4o-mini-FT` is Phrase's fine-tuned version of the `gpt-4o-mini` model.

Engine	F1
`gpt-4o-mini`	0.1835
`gpt-4.1-mini*`	0.2277
`gpt-4o-mini-FT**`	0.2669

We used the Google-MQM test set as our development dataset, as described in Section 2.1.1.

We used GPT engines for this task. We experimented with both the off-the-shelf GPT models, as well as our fine-tuned (FT) versions of some of the models. Fine-tuning was performed on ∼100k segments of Phrase internal data. Table 4 depicts the performance increase when FT is used.

We also experimented with the wording mainly reducing the prompt from the complete MQM task to ESA, i.e., stripping off the error category part of the prompt. We are not allowed to disclose the full prompt but we have experienced that this reduction did not change the performance on our dev set. We have therefore decided to use one of our AutoLQA systems based on the `gpt-4o-mini-FT` model as our primary submission (denoted by double asterisk in Table 4).

Table 5: Task 3 Evaluation results for different systems. C-QE-Δ denotes the COMET-QE delta.

System	C-QE-Δ	TER	GtE-ratio
prod*	0.005	44.51	0.0001
onlyerrors	0.026	25.24	0.0010
accuracy	0.024	15.23	0.0016
fluency	0.050	37.05	0.0014
fluency_s**	0.043	28.27	0.0015

Table 6: Task 3 Primary system quality estimation scores. C-QE denotes the absolute COMET-QE score.

Dataset	baseline		post-edit	
	C-QE	QPS	C-QE	QPS
devtest	0.287	0.901	0.330	0.921
eval-full	0.055	0.878	0.073	0.901
eval-en-cs	0.079	0.910	0.102	0.923

Our secondary submission (denoted by *) is based on the FT gpt-4.1-mini. Due to higher cost, we have not evaluated the model on our development set and have relied on the results in Table 4 and on internal observations at Phrase where in most cases gpt-4.1-mini outperforms gpt-4o-mini.

4 Task 3 (Phrase)

For Task 3, our submissions are based on GPT models as well. We experimented with varying prompts, OpenAI models and settings (notably the amount of reasoning effort which can be set to "low", "medium" or "high" in the API).

After visual inspection and analysis of the error spans and scores provided for the development set, we deemed them quite noisy and not informative enough. We therefore opted for not using this information in the experiments. Our systems only utilise the source sentences and the MT outputs.

Our secondary submission prod (denoted by a single asterisk in Table 5) is loosely inspired by Phrase's production systems for automated MT adaptation. We submitted it as a baseline; however, this system is tailored to addressing typical customer use cases, such as correcting terminology, formality, or the placement of inline tags. Therefore, most of its instructions are not applicable to the WMT setting, so the system wastes its attention on irrelevant aspects. In addition, it required us to set a specific formality and without further domain-specific adjustments, the system likely changed the tone in many cases, leading to substantial amounts of post-editing.

We also experimented with dedicated prompts and a simpler setup targeted towards only improving the general quality of the translations. We simplified the system as well—whereas our production systems typically perform multiple passes over the data, here, we limited the system to just a single pass.

In terms of OpenAI models, we find that reasoning models are well suited for this task. We evaluated several combinations of reasoning effort settings and model types (notably o3 and o3-mini). Our primary submission, as well as most of the contrastive runs, utilise the o3[6] model with medium reasoning effort.

We evaluated four different prompts:[7]

- onlyerrors — focused on finding errors in the translation
- accuracy — improving translation accuracy
- fluency — improving translation fluency
- fluency_s — improving translation fluency, the prompt contains individual steps that the system should follow.

Based on the results on the WMT development set, we selected the prompt fluency_s as our primary submission. This system has the second-highest improvement in COMET-QE score and makes fewer edits than fluency, thereby reaching a better GtE-ratio (gain-to-edit ratio, i.e., the difference in COMET[8] divided by TER, as defined in the task description). It seems noteworthy that COMET-QE score grows more when the prompt is focused on improving translation fluency, as opposed to accuracy.

4.1 Analysis

Based on the official leaderboard, none of our systems improved over the baseline translations on the current test set. In order to explain this negative result, we carried out a post-submission analysis using the current evaluation data.

We found that based on COMET-QE, our system does improve the general translation quality.

[6]The full model name is o3-2025-04-16.
[7]We describe each in more detail in Appendix C.
[8]Approximated here using COMET-QE.

However, we also found COMET-QE to be quite unstable (see the absolute scores across different datasets in Table 6) and decided to complement it with our proprietary QPS metric (see Sec. 2.1.2 for reference). This again showed a consistent improvement in quality.

Finally, we carried out a blinded human evaluation using a small random sample of 50 English-Czech evaluation examples. We removed examples with very low edit rates (TER<10) prior to the sampling in order to prevent the annotator from spending time comparing very similar outputs. The annotator saw each candidate translation side by side (ordered randomly; one being the original MT output and the other our post-edited version) and rated each output on the scale of 1 to 10.

Consistently with the automated quality estimation results (COMET-QE and QPS), the annotator rated the post-edited outputs as higher quality. Out of the 50 examples, our system output was preferred in 31 cases and tied in 10, whereas the baseline output was only preferred in 9 instances. The average score was 8.48±1.20 and 7.50±1.53 (approximate randomization test p-value≈0.002) for the system and baseline respectively, illustrating that while translation quality was quite high overall, translations were not generally deemed perfect by the annotator.

These findings leave conclusions quite open; on the one hand, both automated QE metrics and the human annotator (albeit on just a single language pair) showed clear preference for our post-edits. On the other hand, the official evaluation (which uses reference-based COMET) does not rate the system as better than the baseline. Further analysis leveraging also the official reference translations would be needed to explain this apparent contradiction.

5 Conclusion

We presented our submissions to WMT25 MT Evaluation Task developed by two teams, CUNI and Phrase: two primary and four secondary submissions for Task 1, one primary and two secondary submissions for Task 2 and one primary and one secondary submission for Task 3.

For system fusion/ensembling used in Task 1, the rule "more features are better" generally holds, although not for every individual language-pair or metric. It turns out that the ability of the regressor model "saturates" and gets stuck in the diminishing-reward paradigm soon.

It is interesting to see that competitive results can be achieved by combining very few systems if we choose them well, as is the case of "slick" systems.

In Task 3, we failed to improve on the baseline translation quality in the official rankings. However, our internal post-submission analysis showed a different picture and we currently lack a good explanation for the disagreement.

During the submission, we saw few puzzling outliers for certain language pairs in all tasks' leaderboard, and we expect to reveal the issues after the official annotations are released.

6 Acknowledgment

This work has received funding from the Project OP JAK Mezisektorová spolupráce Nr. CZ.02.01.01/00/23_020/0008518 named "Jazykověda, umělá inteligence a jazykové a řečové technologie: od výzkumu k aplikacím."

References

L. Breiman, Jerome H. Friedman, Richard A. Olshen, and C. J. Stone. 1984. Classification and regression trees.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Hyung Won Chung, Thibault Févry, Henry Tsai, Melvin Johnson, and Sebastian Ruder. 2021. Rethinking embedding coupling in pre-trained language models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Daniel Deutsch, George Foster, and Markus Freitag. 2023. Ties matter: Meta-evaluating modern metrics with pairwise accuracy and tie calibration. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12914–12929, Singapore. Association for Computational Linguistics.

David Freedman. 2005. *Statistical Models : Theory and Practice*. Cambridge University Press.

Markus Freitag, George Foster, David Grangier, Viresh Ratnakar, Qijun Tan, and Wolfgang Macherey. 2021. Experts, errors, and context: A large-scale study of

human evaluation for machine translation. *Transactions of the Association for Computational Linguistics*, 9:1460–1474.

Jerome H. Friedman. 2000. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

Arthur E Hoerl and Robert W Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. DSPy: Compiling declarative language model calls into self-improving pipelines.

Tom Kocmi, Vilém Zouhar, Eleftherios Avramidis, Roman Grundkiewicz, Marzena Karpinska, Maja Popović, Mrinmaya Sachan, and Mariya Shmatova. 2024. Error span annotation: A balanced approach for human evaluation of machine translation. In *Proceedings of the Ninth Conference on Machine Translation*, pages 1440–1453, Miami, Florida, USA. Association for Computational Linguistics.

Arle Richard Lommel, Aljoscha Burchardt, and Hans Uszkoreit. 2013. Multidimensional quality metrics: a flexible system for assessing translation quality. In *Proceedings of Translating and the Computer 35*, London, UK. Aslib.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. 2024. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9340–9366, Miami, Florida, USA. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

F. Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Ole Tange. 2025. Gnu parallel 20250622. GNU Parallel is a general parallelizer to run multiple serial command line programs in parallel without changing them.

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

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, et al. 2024. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1393–1412.

Vilém Zouhar, Shuoyang Ding, Anna Currey, Tatyana Badeka, Jenyuan Wang, and Brian Thompson. 2024. Fine-tuned machine translation metrics struggle in unseen domains. *arXiv preprint arXiv:2306.07899*.

A Task 1 Language-Pair Specific Breakdown

Table 7: Task 1: Language-pair specific breakdown for the submitted systems

		google-mqm		wmt22-qe	
Submission	LP	Kendall's τ	Pearson r	Kendall's τ	Pearson r
phrase-slick	en-de	0.3282	0.6240	0.3089	0.4660
	en-es	0.3293	0.6466	—	—
	en-ru	0.4037	0.6408	0.3571	0.4885
	en-zh	0.2602	0.5467	—	—
	he-en	0.3099	0.4841	—	—
	ja-zh	0.1878	0.3163	—	—
	zh-en	0.2767	0.5971	0.2184	0.3463
palqa-wmt25	en-de	0.2867	0.4886	0.3005	0.3962
	en-es	0.3123	0.5699	—	—
	en-ru	0.3323	0.4341	0.3525	0.4405
	en-zh	0.2105	0.4201	—	—
	he-en	0.3300	0.4829	—	—
	ja-zh	0.2460	0.4050	—	—
	zh-en	0.2502	0.4458	0.2180	0.2636
collab-slick	en-de	0.3232	0.6156	0.3307	0.4805
	en-es	0.3337	0.6417	—	—
	en-ru	0.4016	0.6394	0.3783	0.5011
	en-zh	0.2532	0.5398	—	—
	he-en	0.3189	0.4903	—	—
	ja-zh	0.1964	0.3016	—	—
	zh-en	0.2843	0.6013	0.2207	0.3493
cuni-mr6-overall	en-de	0.1608	0.3648	0.3736	0.5124
	en-es	0.2667	0.4966	—	—
	en-ru	0.2549	0.4706	0.3850	0.4754
	en-zh	0.1142	0.3932	—	—
	he-en	0.2746	0.4372	—	—
	ja-zh	0.2220	0.2402	—	—
	zh-en	0.1632	0.3069	0.1540	0.1889
collab-full	en-de	0.3638	0.6730	0.3235	0.4525
	en-es	0.3486	0.6209	—	—
	en-ru	0.4562	0.7262	0.3790	0.5264
	en-zh	0.2471	0.5099	—	—
	he-en	0.2773	0.4692	—	—
	ja-zh	0.1614	0.2578	—	—
	zh-en	0.3371	0.6908	0.1906	0.3614

B Task 1 - DSPy Models Optimized Prompts

DSPy's resulting "programs" for our *mr** submissions, based on which the final single prompt requesting all the scales or the individual prompts are constructed. These include the selected n-shot examples and any adjustments to the initial data field descriptions or instructions.

B.1 mr6

The full version of optimised "program" (instructions and selected few-shot examples) for *mr6* can be found in the GitHub repository.[9]

B.2 mr7.2.1

The full version of optimised "program" (instructions and selected few-shot examples) for *mr7.2.1* can be found in the GitHub repository.[10]

C Task 3 Prompts

In this section, we share the details of instructions specifically tailored for our WMT systems.

There is also a common core part of the prompt which sets up the task and specifies processing requirements. This part is proprietary and therefore not shared here.

- `onlyerrors` Focus only on errors (grammar, fluency, mistranslation etc.), do not perform subjective or preferential edits.
- `fluency` Concretely, the task is to improve the overall fluency and naturalness (with translation accuracy being important but secondary), while minimising the number of edit operations. Avoid translationese and prefer loose translation when it allows for a more idiomatic translation.
- `fluency_s` Concretely, the task is to improve the overall fluency and naturalness (with translation accuracy being important but secondary), while minimising the number of edit operations. Avoid translationese and prefer loose translation when it allows for a more idiomatic translation.

  Let's go step by step:

  – Identify errors, translationese, or disfluent spans in the translation.
  – Consider several possible corrections.
  – Out of the possible outputs, choose one that is the most natural-sounding and requires few changes (edit operations).
- `accuracy` Concretely, the task is to improve the overall accuracy (with translation fluency being secondary), while minimising the number of edit operations.

[9] https://github.com/hrabalm/wmt25-mt-eval-task/blob/main/mr6/best.json
[10] https://github.com/hrabalm/wmt25-mt-eval-task/blob/main/mr7.2.1/best.json