Diagnose, Then Repair: A Two-Stage MQM-Guided Post-Editing Framework for Domain-Specific Machine Translation

Ji Hun Wang
Amazon
jihunw@amazon.com

Siyu Wu
Amazon
siyumw@amazon.com

Abstract

LLM-based machine translation evaluation can closely match human judgments, but in practice it remains largely diagnostic, with the signals rarely translating into direct quality improvements under real production constraints. We propose a two-stage, evaluator-guided automatic post-editing framework that turns MQM-style evaluation into targeted repairs: a retrieval-augmented LLM evaluator outputs structured, span-level MQM diagnoses under an explicit edit contract, and a separate LLM post-editor applies minimal edits restricted to those diagnoses. This separation improves controllability and reduces paraphrastic drift compared to one-stage "judge-and-refine" baselines. In a systematic study involving seven LLMs spanning three model providers and seven languages, our best configuration consistently improves both COMET-22 and COMETKIWI scores over one-stage post-edit methods, while the evaluator's error spans and severities show strong agreement with human MQM annotations and human editor preferences.

1 Introduction

Machine Translation (MT) systems deployed in specialized domains face recurring failures driven by domain mismatch and domain-specific constraints such as terminology, rare or novel expressions, and style conventions (Koehn and Knowles, 2017). In production settings, quality improvements must also crucially respect fixed compute and latency budgets and operational constraints, making frequent re-training or large-scale re-deployments challenging and difficult to justify economically. Domain distributions are also rarely static, for content refreshes and vocabulary churn can introduce new terms or deprecate old ones, which can quickly erode domain faithfulness even when general MT quality remains strong (Koehn and Knowles, 2017).

Instruction-following LLMs have emerged as an effective solution in this context, for they can often adapt translation behavior via prompting with in-context learning (ICL) and demonstrate competitive translation quality (Jiao et al., 2023; Alves et al., 2023; Wang et al., 2023). However, domain MT still requires evaluation signals aligned with human expectations. Simple algorithmic scores, such as BLEU, are often insufficient and unreliable metrics, missing many adequacy- and terminology-related issues (Mathur et al., 2020), while learned metrics such as COMET-22 correlate better with human judgments (Rei et al., 2020) but inherently cannot localize what should be fixed.

This has motivated the use of descriptive rubrics, such as Multidimensional Quality Metrics (MQM) that provides a structured taxonomy with error category and severity, thereby enabling interpretable error analysis and actionable diagnosis (Lommel et al., 2013, 2014). Recent work has shown that LLMs can be prompted to produce MQM-style error spans directly (Kocmi and Federmann, 2023) and that rubric-style prompting can further improve span-level LLM judgment for high-end models (Kim, 2025). Moreover, MQM-derived feedback can be used to guide LLM post-editing of MT outputs (Ki and Carpuat, 2024; Madaan et al., 2023).

A gap in this thread of research is that self-refinement is difficult to control, and that the resulting gains could often be inconsistent depending on the models and sensitive to prompt design and task setup (Madaan et al., 2023). Furthermore, studies of intrinsic self-correction find that LLMs often fail to reliably detect and fix their own mistakes without external feedback, and repeated self-correction can degrade performance (Huang et al., 2024).

To that end, we propose a modified two-stage MT output repair framework that separates post-edit LLMs from evaluator LLMs, preserving the ben-

efits of MQM-style annotations and error analysis while allowing for directly targeted, controlled fixes while circumventing LLM self-correction.

Our key contributions are as follows:

- We propose a **two-stage MT post-edit framework** that evaluates first and then post-edits. We empirically validate its effectiveness compared to one-stage approach.
- We investigate how different prompting strategies could affect post-edit performance, and show that our prompt that incorporates RAG and "edit contract" (Section 3.2) generates better post-edits.
- Through a systematic study involving seven models spanning three model providers and various model sizes, we assess how model diversity and model sizes could affect the post-edit performance, which could be used as general guidance when adopting the framework.

2 Related Work

MQM-based evaluation and LLM annotators. MQM is a fine-grained framework for MT quality assessment that organizes errors into a flexible taxonomy including error labels and severity, making evaluation actionable beyond single-score metrics (Lommel et al., 2013, 2014). It has also been used in large-scale expert human evaluation setups (Freitag et al., 2021) and as a basis for meta-evaluation in WMT (Freitag et al., 2022). "LLM-as-a-judge or jury" work has operationalized MQM with structured prompting, so that LLMs can produce rubric-faithful MQM judgments, including span-level error localizations and reasoning (Zheng et al., 2023; Liu et al., 2023; Chiang and Lee, 2023; Verga et al., 2024). Works such as GEMBA-MQM (Kocmi and Federmann, 2023) and RUBRIC-MQM (Kim, 2025) build on this thread to make the approach more reliable and aligned with human judgments.

Post-editing translations with LLMs. Automatic post-editing of MT outputs using strong instruction-following LLMs has garnered attention as an effective approach to directly improve the translations without re-training the MT model (Chatterjee et al., 2018; Simard et al., 2007). It has been shown that a single-shot post-edit or iterative refinement of edits that alternate assessment and revision are both effective (Chen et al., 2024; Wu et al., 2025; Briakou et al., 2024). A closely related thread explicitly couples quality estimation or structured error feedback with post-editing, motivating pipelines where an evaluator's diagnostic signal guides subsequent edits rather than relying on generic instructions (Ki and Carpuat, 2024; Lu et al., 2024).

Retrieval-augmented techniques for domain MT. Retrieval-augmented generation (RAG) is a technique that incorporates relevant information from external sources to MT models (such as seq-2-seq models or LLMs) to boost their domain-specific knowledge (Lewis et al., 2021). Existing work on domain MT has leveraged paired sentence exemplars retrieved from an external source as a method of data augmentation or decoding heuristics (Zhang et al., 2018; Bulte and Tezcan, 2019; Khandelwal et al., 2021) or a multilingual knowledge graphs using domain-specific examples (Conia et al., 2024; Wang et al., 2025), to guide the model on terminological or stylistic elements specific to the domain.

3 Method

3.1 Overview

We study **segment-level domain-specific MT**. Given a source segment x, an MT model produces an initial hypothesis $y^{(0)}$ in the target language, where the ground truth target segment is y^*. We aim to produce improved translations $y^{(i)}$ for the iteration count $i = 1, 2, \cdots$, via an LLM-guided, retrieval-grounded post-edit step, dispensing with the costly MT model re-training.

Our design is intentionally modular, enforcing a strict separation of burden between the evaluator and post-editor: after the generation of $y^{(0)}$, an LLM evaluator first produces structured, MQM-style diagnoses, and post-editor LLMs sequentially apply targeted repairs based on the evaluator's error analysis and edit suggestion. This design avoids self-correction with automated feedback, which has been shown a challenging task particularly for complex reasoning tasks (Tie et al., 2025; Pan et al., 2023; Madaan et al., 2023), and can also suppresses overcorrection, as the post-editors are not asked to broadly "improve" translations, which could degrade the performance, but asked to resolve specific, diagnosed issues. Another benefit of our design is that it gives flexibility to apply additional post-editing steps as needed and easily pair different models depending on use cases and upon newer model releases.

Our system orchestrates retrieval, evaluation, correction, and stopping, imposed via prompts and hard constraints. Specifically, it:

1. retrieves k semantically similar segments from the domain knowledge base (KB) $\mathcal{D}$;
2. evaluates once to obtain the MQM-style report r, and stops if no major or minor issues are reported;
3. otherwise applies additional post-editing steps as needed.

In subsequent subsections, we elaborate on the responsibilities and behaviors of the evaluator and post-editors in a precise way.

3.2 Edit Contracts

In our design of the system, we identify that the separation of evaluator and post-editor responsibilities may give rise to complications, as delegating the post-editing duty to a separate model may bring about undesired consequences. To mitigate such concern, we introduce **edit contracts** as a section of the prompt, bridging the gap between the evaluator's requests and post-editor's requirements.

As part of the edit contract, the evaluator provides an explicit guidance on suggested fixes on each issue type, and post-editors are required to act based on the specific terms. The expectation is that the post-editors are able to focus on generating edits that are attributed to evaluator issues and/or retrieved evidence, hence effectively decreasing the likelihood of overcorrection. We provide our form of edit contract in Appendix E.

3.3 Structured MQM Evaluator

An evaluator E is an LLM responsible for analyzing $y^{(0)}$ and generating r in a reference-free manner. The prompt given to E is comprised of the static portion p_E (minimally including the task description, MQM rubric, output schema, and evaluator-side edit contract) and dynamic information that differs per query (such as x, $y^{(0)}$, and k semantically most similar exemplars $\{s_i\}_{i=1}^{k}$ retrieved from the KB $\mathcal{D}$ populated with high-quality source-target segment pairs for the purpose of providing domain-specific knowledge to the evaluator). Formally, we have:

$$r = \{e_j\}_{j=1}^{n} = E(p_E, x, y^{(0)}, \{s_i\}_{i=1}^{k}),$$

where e_j denotes an analysis of the diagnosed issue, generated based on the required schema, which contains fields such as `issue_id`, `span`, `category`, `severity` $\in$ {`major`, `minor`, `neutral`}, and a short `description`, optionally with fields such as `suggested_fix` and `evidence_ids` linking to retrieved items.

In our system, we run the evaluator only once per query, producing r for the initial hypothesis $y^{(0)}$.

3.4 Post-Editors

If r contains any issues, we send the evaluator output and $y^{(0)}$ to the post-editors. Each post-editor outputs a new hypothesis $y^{(i)}$ and an edit log $\ell^{(i)}$ describing each change made, which should reference one or more `issue_id`'s. We explicitly instruct the post-editors by including a statement within its edit contract section of the static prompt p_C that they should prioritize minimal edits, avoiding paraphrases of unaffected spans. To formalize, a post-editor C_i is an LLM that receives a 4-tuple $(p_C, x, r, \{y^{(j)}\}_{j=0}^{i-1})$ as part of its prompt, where p_C denotes the static task description for the post-editor and $\{y^{(j)}\}$ a compact history of hypotheses generated up to this point. The post-editor prompt is intentionally concise and constrained to diagnosis-conditioned repairs (see Appendix E), while the inclusion of history enables later post-editors to reference earlier edits and decisions made by the previous set of post-editors.

The modularized aspect of the system also allows for applying the post-editing step more than once, to a sequence of post-editors $\{C_1, C_2, \cdots\}$. For C_i where $i \geq 2$, it first assesses whether $y^{(i-1)}$ resolves all errors. If it judges so, it immediately exits; otherwise, it generates a 2-tuple $(y^{(i)}, \ell^{(i)})$, and potentially continues iterating afterwards.

4 Experiments

4.1 Experimental Setup

Data. We evaluate on internal Translation Memory representing the e-commerce domain, with 5k random samples as the held-out stratified test set and $\mathcal{D}$ built from non-test segments. Full dataset details are available in Appendix D.

Languages. We focus on translating English (EN) segments to six languages: German (DE), Italian (IT), Spanish (ES), French (FR), Chinese (CN), and Japanese (JP).

Models. We use Anthropic's Claude 3.5 Sonnet and Claude Opus 4.5 as our MT model generating $y^{(0)}$ and our evaluator, respectively. For post-editors, we experiment with a diverse set of LLMs available on Amazon Bedrock, spanning multiple providers and sizes. For a complete list of LLMs, refer to Appendix D. We use Amazon OpenSearch Service for KB construction and retrieval.

Evaluation. We report the quality of generated translations using Unbabel's reference-based evaluation metric COMET-22 (Rei et al., 2022a) and reference-free evaluation metric COMETKIWI (Rei et al., 2022b), together of which can provide a holistic picture of how our system performs.

4.2 Experimental Design

The following steps delineate the end-to-end process of our experiments:

1. The MT model generates $y^{(0)}$, which is passed to the MQM evaluator. The evaluator prompt, illustrated in Appendix E, is built on GEMBA-MQM but further enhanced by incorporating RAG, including a descriptive rubric, and specifying the edit contract (Section 3.2).

2. In the first round of post-editing, each post-editor in our LLM universe analyzes $y^{(0)}$, referencing the evaluator output, and generates $y^{(1)}$. This process yields the post-editor C_1^* that makes the biggest improvement of post-edit quality on average. We note that this is a fair comparison, for $y^{(0)}$ and the evaluator output are shared across all post-editors.

3. In the second round of post-editing, we pass the output of C_1^* to the same set of LLMs, where each post-editor can either conclude all errors have been addressed and therefore exit, or flag the presence of errors and post-edit $y^{(1)}$ to generate $y^{(2)}$.

4. We can optionally iterate this process by feeding the output of C_i^* and evaluator output to C_{i+1}'s.

4.3 Experiment Variations

To assess the effectiveness of our prompt and utilizing separate post-editors in the first place, we propose the following set of additional experiments differing in terms of the evaluator prompts:

P1. GEMBA-MQM prompt (Kocmi and Federmann, 2023);

P2. GEMBA-MQM prompt + descriptive rubric + edit contract;

P3. GEMBA-MQM prompt + descriptive rubric + edit contract + RAG.

Furthermore, to investigate the effectiveness of separating evaluators and post-editors, we propose running an additional experiment that combines evaluators and post-editors (*i.e.*, having Claude Opus 4.5 analyze the errors contained within $y^{(0)}$ and also generate post-edits based on them), utilizing the best prompting strategy among **P1** through **P3** as empirically validated:

P4. Empirically the best prompting strategy on average from **P1** to **P3** + post-edit instructions.

5 Results and Analysis

In this section, we mainly focus on the results for EN → DE and EN → CN. However, our findings generalize across all six languages, demonstrating the effectiveness of our proposed approach. The full results can be found in Appendix A.

5.1 Which Prompt Works the Best?

The primary question concerns how different evaluator prompting strategies affect downstream post-edit quality. We compare three variants (Section 4.3): **P1** (GEMBA-MQM), **P2** (MQM + descriptive rubric + edit contract), and **P3** (**P2** + RAG with domain exemplars).

Table 1 summarize COMET-22 and COMETKIWI scores for $y^{(1)}$. Compared to **P1**, **P2** yields small but consistent improvements, indicating that descriptive rubrics and structured edit contracts improve the utility of evaluator outputs. We observe the largest gains under **P3**, where the retrieval from $\mathcal{D}$ further improves both metrics (notably for EN → DE). This suggests that domain exemplars can benefit the generated post-edits by reducing ambiguity in terminology and style, enabling more precise diagnoses and better-aligned edits. Importantly, these gains are consistently observed under a fixed post-editor LLM (Claude Sonnet 4.5), confirming that the evaluator design directly influences repair quality. Overall, structured grounding and constrained edit guidance outperform unconstrained refinement, improving post-edit quality while reducing unnecessary rewrites.

Method	Evaluator Prompt	Post-editor	EN → DE		EN → CN	
			COMET	COMETKIWI	COMET	COMETKIWI
No post-edit	–	–	86.39	82.66	87.76	81.60
One-stage	**P3**	Claude Opus 4.5	89.38 (+2.99)	82.84 (+0.18)	88.70 (+0.94)	79.57 (-2.03)
Two-stage	**P1**	Claude Sonnet 4.5	86.47 (+0.08)	82.65 (-0.01)	87.97 (+0.21)	81.70 (+0.10)
Two-stage	**P2**	Claude Sonnet 4.5	86.57 (+0.18)	82.66 (+0.00)	88.86 (+1.10)	81.67 (+0.07)
Two-stage	**P3**	Claude Sonnet 4.5	**89.93** (+3.54)	**84.50** (+1.84)	**90.80** (+3.04)	**83.19** (+1.59)

Table 1: COMET-22 and COMETKIWI scores of $y^{(1)}$ using one-stage and two-stage post-edit method, the latter with three prompt varieties; **P1** is GEMBA-MQM prompt, **P2** is MQM + rubric + edit contract, and **P3** is MQM + rubric + edit contract + RAG. In this table, we have selected EN → DE and EN → CN for display. For complete results across all six target languages, refer to Appendix A.

5.2 Is a separate post-edit process effective?

Table 2 (and the full results in Table 5 in the Appendix A) shows that evaluator-guided post-editing helps, as measured by COMET-22 score improvements compared against the no post-edit baseline, and that the two-stage design is generally more effective than a one-stage "judge-and-refine" approach: 4 out of 7 models yield a higher COMET score for EN → DE, while 6 out of 7 models yield a higher COMET-22 score for EN → CN. Based on the substantial gains obtained by the two-stage method for DE and CN, we conclude that the two-stage post-edit process with our prompt design (Section 4.3 and Appendix E) systematically translates into measurable quality improvements without re-training the underlying model.

Overall, these results support our central claim that there is a clear benefit derived from the separation of evaluators and post-editors, reducing unconstrained re-writing and better preserving already correct spans, thereby yielding better outputs overall while maintaining all the benefits that one-stage post-edit framework offers.

5.3 Which LLM generates the best post-edits?

Our LLM universe spans three model providers: Google, Anthropic, and OpenAI. From each model provider, we have carefully selected models on both ends of the spectrum in terms of model sizes, as measured by the number of parameters. This allows us to analyze how post-edit qualities differ across model providers and sizes.

From Table 2, we find that the Claude models show the strongest post-edit capability for EN → DE language direction, with Claude 3.5 Haiku showing higher COMET-22 scores than both GPT OSS models and similar scores compared to Gemma 3 27B model. COMETKIWI scores also support a similar conclusion.

For EN → CN, Gemma 3 is the overall winner, yielding higher COMET-22 scores compared to Claude and GPT OSS models. Interestingly, while Gemma 3 4B shows the smallest COMET score gain among all considered LLMs for EN → DE and even lower COMETKIWI score than that of $y^{(0)}$, we find that it achieves the highest average COMET score for EN → CN. Larger models tend to show better post-edit qualities, with one notable exception being Gemma 3 4B for EN → CN. This observation suggests that there is not a single best model, making it necessary to experiment with a large universe of models with the data representative of the population prior to selecting the model for a specific language direction and domain.

5.4 How often does C_2 think $y^{(1)}$ still contains errors?

To quantify how much benefit a second post-editing round could yield (that is, E, followed by C_1, followed by C_2; refer to Section 3.4), we measure how often C_2 flags the first-round output $y^{(1)}$ (generated by Claude Sonnet 4.5) as still containing errors. Table 3 reports results for EN → DE.

C_2 Model	# Flagged	Rate (%)
Claude 3.5 Haiku	205 / 4031	5.1
Claude Sonnet 4	875 / 4031	21.7
Claude Sonnet 4.5	1282 / 4031	31.8
Gemma 3 4B	328 / 4031	8.1
Gemma 3 27B	541 / 4031	13.4
GPT OSS 20B	278 / 4031	6.9
GPT OSS 120B	207 / 4031	5.1

Table 3: Continuation rate of C_2 after first-round correction ($y^{(1)}$) for EN → DE.

Method	Model	EN → DE		EN → CN	
		COMET	COMETKIWI	COMET	COMETKIWI
No post-edit	–	86.39	82.66	87.76	81.60
One-stage	Claude Opus 4.5	89.38 (+2.99)	–	88.70 (+0.94)	–
Two-stage	Gemma 3 4B	88.29 (+1.90)	82.43 (-0.23)	**91.08** (+3.32)	79.82 (-1.78)
	Gemma 3 27B	89.62 (+3.23)	83.03 (+0.37)	90.83 (+3.07)	81.92 (+0.32)
	Claude 3.5 Haiku	89.53 (+3.14)	82.69 (+0.03)	90.20 (+2.44)	82.51 (+0.91)
	Claude Sonnet 4	89.85 (+3.46)	83.45 (+0.79)	90.56 (+2.80)	82.94 (+1.34)
	Claude Sonnet 4.5	**89.93** (+3.54)	**84.50** (+1.84)	90.80 (+3.04)	**83.19** (+1.59)
	GPT OSS 20B	89.10 (+2.71)	82.67 (+0.01)	87.98 (+0.22)	79.06 (-2.54)
	GPT OSS 120B	89.18 (+2.79)	82.76 (+0.10)	89.63 (+1.87)	79.52 (-2.08)

Table 2: COMET-22 and COMETKIWI scores of $y^{(0)}$ and $y^{(1)}$ across all considered LLMs, using **P3** as the evaluator prompting strategy

Across 4,031 segments, most models judge the majority of first-round outputs as sufficiently corrected. For several models (Claude 3.5 Haiku, GPT OSS 120B), only around 5% of segments are flagged for further editing, suggesting that a single targeted repair pass effectively resolves most diagnosed issues. In contrast, stronger Claude model variants tend to apply stricter criteria (Claude Sonnet 4 flags 21.7% of segments, and Claude Sonnet 4.5 flags 31.8%), indicating that higher-capacity reasoning models are more likely to detect residual minor or stylistic issues even after targeted post-edits.

Overall, these results suggest that one round of post-edit process is likely sufficient for the majority of segments, while additional rounds primarily serve to address stricter quality thresholds rather than widespread unresolved major errors.

5.5 Are second round post-edits beneficial?

We next evaluate whether applying post-editing with C_2 yields further quality gains over the first-round output $y^{(1)}$. Table 4 reports EN → DE results using the best-performing prompt configuration (**P3**). Claude Sonnet 4.5 serves as C_1, and we consider all other models in our universe as C_2.

The results show that a second post-editing round generally degrades post-edit quality: in six out of seven cases, COMET-22 scores decrease, sometimes substantially (for instance, -1.86 for GPT-OSS-120B). Only when the same high-capacity model (Claude Sonnet 4.5) is used again as C_2 do we observe a marginal improvement ($+0.14$), which is small relative to the first-round gains.

C_2 Model	$y^{(2)}$ COMET
Gemma 4B	89.18 (-0.75)
Gemma 27B	89.93 (+0.00)
Claude 3.5 Haiku	89.63 (-0.30)
Claude Sonnet 4	89.31 (-0.62)
Claude Sonnet 4.5	90.07 (+0.14)
GPT OSS 20B	89.30 (-0.63)
GPT OSS 120B	88.07 (-1.86)

Table 4: Impact of second-round post-editing (C_2) on EN → DE

These findings suggest that most systematic errors are already resolved during the first evaluator-guided post-edit process. Additional rounds tend to introduce paraphrastic drift or unnecessary edits that degrade adequacy or fluency. In practice, a single structured evaluation followed by one targeted post-editing step appears sufficient, and further iterations offer limited benefit while increasing regression risk.

It is also worth noting that there could be a model-specific behavior at play. Combining the analysis from Tables 3 and 4, we find that Claude Sonnet 4.5, which is the excelling post-editor from previous experiments, is also more likely to flag issues even in the potential absence of real errors that should be pointed out. One possible reason for this behavior is the stronger instruction-following abilities obtained by the model as reinforced during the alignment process.

6 Conclusion and Future Work

In this work, we proposed a two-stage post-edit framework for MT in specialized domains. By sep-

arating a retrieval-augmented evaluator from a constrained post-editor governed by edit contracts, the two-stage design delivers consistent gains across domain translation tasks on two language directions. Results from a systematic study involving seven LLMs show that an MQM-style, evidence-grounded evaluation can improve post-editor's correction quality and that a one-round targeted post-edits are sufficient for improving the post-edit quality, with further iteration offering only limited benefits. Future work includes exploring confidence-aware post-editing to further optimize cost-quality tradeoffs, and extending the framework to additional domains and language pairs.

References

Duarte Alves, Nuno Guerreiro, João Alves, José Pombal, Ricardo Rei, José de Souza, Pierre Colombo, and Andre 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.

Eleftheria Briakou, Jiaming Luo, Colin Cherry, and Markus Freitag. 2024. Translating step-by-step: Decomposing the translation process for improved translation quality of long-form texts. *Preprint*, arXiv:2409.06790.

Bram Bulte 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, Florence, Italy. Association for Computational Linguistics.

Rajen Chatterjee, Matteo Negri, Raphael Rubino, and Marco Turchi. 2018. Findings of the WMT 2018 shared task on automatic post-editing. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 710–725, Belgium, Brussels. Association for Computational Linguistics.

Pinzhen Chen, Zhicheng Guo, Barry Haddow, and Kenneth Heafield. 2024. Iterative translation refinement with large language models. In *Proceedings of the 25th Annual Conference of the European Association for Machine Translation (Volume 1)*, pages 181–190, Sheffield, UK. European Association for Machine Translation (EAMT).

Cheng-Han Chiang and Hung-yi Lee. 2023. A closer look into using large language models for automatic evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8928–8942, Singapore. Association for Computational Linguistics.

Simone Conia, Daniel Lee, Min Li, Umar Farooq Minhas, Saloni Potdar, and Yunyao Li. 2024. Towards cross-cultural machine translation with retrieval-augmented generation from multilingual knowledge graphs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16343–16360, Miami, Florida, USA. Association for Computational Linguistics.

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.

Markus Freitag, Ricardo Rei, Nitika Mathur, Chi-kiu Lo, Craig Stewart, Eleftherios Avramidis, Tom Kocmi, George Foster, Alon Lavie, and André F. T. Martins. 2022. Results of WMT22 metrics shared task: Stop using BLEU – neural metrics are better and more robust. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 46–68, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. *Preprint*, arXiv:2310.01798.

Wenxiang Jiao, Wenxuan Wang, Jen tse Huang, Xing Wang, Shuming Shi, and Zhaopeng Tu. 2023. Is chatgpt a good translator? yes with gpt-4 as the engine. *Preprint*, arXiv:2301.08745.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. *Preprint*, arXiv:2010.00710.

Dayeon Ki and Marine Carpuat. 2024. Guiding large language models to post-edit machine translation with error annotations. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4253–4273, Mexico City, Mexico. Association for Computational Linguistics.

Ahrii Kim. 2025. RUBRIC-MQM : Span-level LLM-as-judge in machine translation for high-end models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 147–165, Vienna, Austria. Association for Computational Linguistics.

Tom Kocmi and Christian Federmann. 2023. Gemba-mqm: Detecting translation quality error spans with gpt-4. *Preprint*, arXiv:2310.13988.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *Preprint*, arXiv:1706.03872.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Arle Lommel, Aljoscha Burchardt, Maja Popović, Kim Harris, Eleftherios Avramidis, and Hans Uszkoreit. 2014. Using a new analytic measure for the annotation and analysis of MT errors on real data. In *Proceedings of the 17th Annual Conference of the European Association for Machine Translation*, pages 165–172, Dubrovnik, Croatia. European Association for Machine Translation.

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.

Qingyu Lu, Liang Ding, Kanjian Zhang, Jinxia Zhang, and Dacheng Tao. 2024. Mqm-ape: Toward high-quality error annotation predictors with automatic post-editing in llm translation evaluators. *Preprint*, arXiv:2409.14335.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *Preprint*, arXiv:2303.17651.

Nitika Mathur, Timothy Baldwin, and Trevor Cohn. 2020. Tangled up in BLEU: Reevaluating the evaluation of automatic machine translation evaluation metrics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4984–4997, Online. Association for Computational Linguistics.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2023. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *Preprint*, arXiv:2308.03188.

Ricardo Rei, José G. C. de Souza, Duarte Alves, Chrysoula Zerva, Ana C Farinha, Taisiya Glushkova, Alon Lavie, Luisa Coheur, and André F. T. Martins. 2022a. COMET-22: Unbabel-IST 2022 submission for the metrics shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Ricardo Rei, Marcos Treviso, Nuno M. Guerreiro, Chrysoula Zerva, Ana C Farinha, Christine Maroti, José G. C. de Souza, Taisiya Glushkova, Duarte Alves, Luisa Coheur, Alon Lavie, and André F. T. Martins. 2022b. CometKiwi: IST-unbabel 2022 submission for the quality estimation shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 634–645, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Michel Simard, Cyril Goutte, and Pierre Isabelle. 2007. Statistical phrase-based post-editing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 508–515, Rochester, New York. Association for Computational Linguistics.

Guiyao Tie, Zenghui Yuan, Zeli Zhao, Chaoran Hu, Tianhe Gu, Ruihang Zhang, Sizhe Zhang, Junran Wu, Xiaoyue Tu, Ming Jin, Qingsong Wen, Lixing Chen, Pan Zhou, and Lichao Sun. 2025. Can llms correct themselves? a benchmark of self-correction in llms. *Preprint*, arXiv:2510.16062.

Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. 2024. Replacing judges with juries: Evaluating llm generations with a panel of diverse models. *Preprint*, arXiv:2404.18796.

Jiaan Wang, Fandong Meng, Yingxue Zhang, and Jie Zhou. 2025. Retrieval-augmented machine translation with unstructured knowledge. *Preprint*, arXiv:2412.04342.

Longyue Wang, Chenyang Lyu, Tianbo Ji, Zhirui Zhang, Dian Yu, Shuming Shi, and Zhaopeng Tu. 2023. Document-level machine translation with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16646–16661, Singapore. Association for Computational Linguistics.

Di Wu, Seth Aycock, and Christof Monz. 2025. Please translate again: Two simple experiments on whether human-like reasoning helps translation. *Preprint*, arXiv:2506.04521.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, and Satoshi Nakamura. 2018. Guiding neural machine translation with retrieved translation pieces. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1325–1335, New Orleans, Louisiana. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Appendix

A Complete Results

Method	Model	EN → DE		EN → CN		EN → FR	
		COMET	COMETKIWI	COMET	COMETKIWI	COMET	COMETKIWI
No post-edit	–	86.39	82.66	87.76	81.60	87.19	82.65
One-stage	Claude Opus 4.5	89.38 (+2.99)	82.84 (+0.18)	88.70 (+0.94)	79.57 (-2.03)	**90.83** (+3.64)	**83.29** (+0.64)
Two-stage	Gemma 3 4B	88.29 (+1.90)	82.43 (-0.23)	**91.08** (+3.32)	79.82 (-1.78)	89.36 (+2.17)	82.90 (+0.25)
	Gemma 3 27B	89.62 (+3.23)	83.03 (+0.37)	90.83 (+3.07)	81.92 (+0.32)	90.55 (+3.36)	83.15 (+0.50)
	Claude 3.5 Haiku	89.53 (+3.14)	82.69 (+0.03)	90.20 (+2.44)	82.51 (+0.91)	90.56 (+3.37)	82.97 (+0.32)
	Claude Sonnet 4	89.85 (+3.46)	83.45 (+0.79)	90.56 (+2.80)	82.94 (+1.34)	90.76 (+3.57)	83.22 (+0.57)
	Claude Sonnet 4.5	**89.93** (+3.54)	**84.50** (+1.84)	90.80 (+3.04)	**83.19** (+1.59)	90.81 (+3.62)	83.27 (+0.62)
	GPT OSS 20B	89.10 (+2.71)	82.67 (+0.01)	87.98 (+0.22)	79.06 (-2.54)	89.56 (+2.37)	82.10 (-0.55)
	GPT OSS 120B	89.18 (+2.79)	82.76 (+0.10)	89.63 (+1.87)	79.52 (-2.08)	88.37 (+1.18)	81.29 (-1.36)

Method	Model	EN → ES		EN → IT		EN → JP	
		COMET	COMETKIWI	COMET	COMETKIWI	COMET	COMETKIWI
No post-edit	–	83.01	82.02	87.36	**84.28**	89.08	**84.79**
One-stage	Claude Opus 4.5	89.59 (+6.58)	81.73 (-0.29)	91.25 (+3.89)	82.77 (-1.51)	93.48 (+4.40)	83.96 (-0.83)
Two-stage	Gemma 3 4B	83.73 (+0.72)	80.16 (-1.86)	89.36 (+2.00)	82.11 (-2.17)	89.73 (+0.65)	82.52 (-2.27)
	Gemma 3 27B	89.17 (+6.16)	81.59 (-0.43)	90.72 (+3.36)	82.45 (-1.83)	91.59 (+2.51)	82.92 (-1.87)
	Claude 3.5 Haiku	90.35 (+7.34)	81.75 (-0.27)	90.64 (+3.28)	82.10 (-2.18)	92.43 (+3.35)	83.06 (-1.73)
	Claude Sonnet 4	90.91 (+7.90)	82.05 (+0.03)	91.02 (+3.66)	82.57 (-1.71)	92.82 (+3.74)	83.79 (-1.00)
	Claude Sonnet 4.5	**90.93** (+7.92)	**82.12** (+0.10)	**91.05** (+3.69)	82.63 (-1.65)	**92.87** (+3.79)	83.80 (-0.99)
	GPT OSS 20B	87.30 (+4.29)	79.00 (-3.02)	89.79 (+2.43)	81.29 (-2.99)	91.71 (+2.63)	82.71 (-2.08)
	GPT OSS 120B	87.44 (+4.43)	78.99 (-3.03)	88.82 (+1.46)	80.72 (-3.56)	90.83 (+1.75)	82.12 (-2.67)

Table 5: COMET-22 and COMETKIWI scores of $y^{(0)}$ and $y^{(1)}$ across all considered LLMs, using **P3** as the evaluator prompting strategy

The complete results across all six language directions, reported in Table 5, confirm that our findings from EN → DE and EN → CN generalize broadly. Evaluator-guided post-editing consistently improves COMET-22 scores over the no-post-edit baseline regardless of target language, and the two-stage design outperforms the one-stage baseline in the majority of cases across all directions. EN → FR is a notable anomaly where one-stage post-edit yields the best result over all two-stage results, although the gap between the one-stage performance and the best two-stage performance (Claude Sonnet 4.5) is minimal.

Claude Sonnet 4.5 remains the strongest post-editor overall, achieving the best or near-best COMET-22 scores in most language directions, while the Claude family models in general demonstrate strong and consistent gains across European languages. The GPT OSS models yield competitive COMET-22 improvements for European target languages but show more variable COMETKIWI behavior, echoing the pattern observed in the main results. These results further support the recommendation that practitioners should experiment with a wide range of models for their specific languages of interest and domains prior to selecting a post-editor for deployment.

B Ablation Studies and Further Investigation of the System

B.1 Varying Evaluator LLMs

In the main experiments, we reserved Claude Opus 4.5 as our evaluator LLM. A reasonable question that arises concerns how the system's performance shifts when we utilize different evaluators. To investigate this effect, we conducted an ablation study on EN $\rightarrow$ FR, varying the evaluator model while keeping the post-editor (Claude Sonnet 4.5) and the prompt (**P3**) fixed. The results are as follows:

Evaluator LLM	COMET	COMETKIWI
No post-edit	87.19	82.65
Claude 3.5 Haiku	88.28 (+1.09)	**83.97** (+1.32)
Claude Sonnet 4	89.80 (+2.61)	83.59 (+0.94)
Claude Sonnet 4.5	90.37 (+3.18)	83.10 (+0.45)
Claude Opus 4.5	**90.81** (+3.62)	83.27 (+0.62)

Table 6: COMET-22 and COMETKIWI scores of $y^{(1)}$ upon varying the evaluator LLM

We observe that even the smallest and most cost-efficient evaluator (Claude 3.5 Haiku) achieves a COMET-22 score of 88.28, a $+1.09$ improvement over the no post-edit baseline. This gain is obtained using an evaluator that is substantially cheaper than Claude Opus 4.5, demonstrating that the framework does not critically depend on the most powerful evaluator to deliver improvements. As the evaluator model scales up, COMET-22 increases monotonically, indicating that stronger evaluators do contribute additional gains, but the framework remains effective across the full range of evaluator capabilities.

B.2 Comparison Against GEMBA-MQM

For our main set of experiments as displayed in Table 1, we tested one-stage framework paired with **P3** as the evaluator prompt. For a fair comparison, we conducted additional experiments to validate our approach against GEMBA-MQM as one-stage post-edit process, where the model is asked to both diagnose and correct errors in a single inference call. The results on EN $\rightarrow$ DE are as follows:

Method	Model	COMET	COMETKIWI
No post-edit	Claude 3.5 Sonnet	86.39	82.66
One-stage (GEMBA-MQM)	Claude Opus 4.5	86.51 (+0.12)	82.88 (+0.22)
Two-stage	Claude Opus 4.5 + Claude Sonnet 4.5	**89.93** (+3.54)	**84.50** (+1.84)

Table 7: COMET-22 and COMETKIWI scores for $y^{(0)}$ and $y^{(1)}$ for one-stage GEMBA-MQM (*i.e.* with prompt **P1**) and our two-stage framework

The one-stage GEMBA-MQM baseline yields only marginal improvements over the original translation ($+0.12$ COMET-22 and $+0.22$ COMETKIWI), despite identifying and attempting to correct 44.8% of segments. This is consistent with prior findings that LLMs struggle to simultaneously diagnose and repair their own outputs in a single pass. In contrast, our two-stage framework achieves substantially larger gains ($+3.54$ COMET-22, $+1.84$ COMETKIWI), validating the core thesis of our proposed framework that decoupling evaluation from post-editing enables more effective translation refinement.

C Latency Analysis

The following table provides the latency statistic involved in the post-edit procedure, computed over 50 segments:

Stage	Mean	Median	P75	P95	Stdev
Evaluator (Claude Opus 4.5)	14.34	7.80	12.54	54.38	17.38
Post-editor (Claude Sonnet 4.5)	3.30	3.10	4.15	5.04	1.08
End-to-end (single round)	17.64	10.90	16.69	59.41	17.41

Table 8: Latency statistics (in seconds) for each stage of the two-stage pipeline, measured over 50 segments

Based on these numbers, we contend that, while the approach may be limited in situations that require instantaneous edits (such as conversational settings), it would still be suitable for situations that can tolerate slight latency for offline evaluation.

In domain-specific machine translation, the content is often highly repetitive. For e-commerce, for example, segments contain a lot of product descriptions, policy text, and shipping notices. As such, a cache keyed on source segment embeddings could bypass the evaluator entirely for segments similar to previously diagnosed ones, delivering an additional advantage of reduced costs. In our dataset, we also observe significant lexical overlap across segments, suggesting that the cache hit rates would be very high in practice.

D List of LLMs and Dataset Details

We compose our universe of post-editor LLMs with models of varying providers and sizes, to have a glimpse of the effect of model diversity and size on the quality of post-edits. We focus on models that support batch inference on Amazon Bedrock as of February 2026. We further note that, for certain models, the information regarding the number of active parameters is not publicly released, in which case we mark the relevant cell with $(*)$. The unit of context window and max output is tokens; the "Pricing" column lists the input and output prices per million tokens with real-time inference on Amazon Bedrock.

Tables 9 and 10 summarize the LLMs and data splits for each language direction we used for the experiments, respectively:

	Model	# Params	Bedrock ID	Context / Max Output	Pricing
Google					
1	Gemma 3 4B IT	4 billion	`google.gemma-3-4b-it`	128,000 / 8,192	$0.04 / $0.08
2	Gemma 3 27B PT	27 billion	`google.gemma-3-27b-it`	128,000 / 8,192	$0.23 / $0.38
Anthropic					
1	Claude 3.5 Haiku	$(*)$	`anthropic.claude3-5-haiku-20241022-v1:0`	200,000 / 8,192	$0.8 / $4
2	Claude Sonnet 4	$(*)$	`anthropic.claude-sonnet-4-20250514-v1:0`	200,000 / 8,192	$3 / $15
3	Claude Sonnet 4.5	$(*)$	`anthropic.claude-sonnet-4-5-20250929-v1:0`	200,000 / 64,000	$3 / $15
4	Claude Opus 4.5	$(*)$	`anthropic.claude-opus-4-5-20251101-v1:0`	200,000 / 64,000	$5 / $25
OpenAI					
1	GPT OSS 20B	20 billion	`openai.gpt-oss-20b-1:0`	128,000 / 16,000	$0.07 / $0.3
2	GPT OSS 120B	120 billion	`openai.gpt-oss-120b-1:0`	128,000 / 16,000	$0.15 / $0.6

Table 9: Model details

	Name	Domain	Language Direction	Split Size	
				KB	Test
1	Translation Memory (TM)	e-commerce	EN $\rightarrow$ DE	135,000	5,000
2			EN $\rightarrow$ IT	140,000	5,000
3			EN $\rightarrow$ ES	184,000	5,000
4			EN $\rightarrow$ FR	141,000	5,000
5			EN $\rightarrow$ CN	170,000	5,000
6			EN $\rightarrow$ JP	152,000	5,000

Table 10: Dataset details

E Prompts

E.1 Evaluator Prompts

Evaluator Prompt (GEMBA-MQM + Rubric + Edit Contract + RAG)

You are an annotator for the quality of machine translation. Your task is to identify errors and assess the quality of the translation, grounded in domain-specific knowledge provided via retrieved exemplars.

Domain exemplars (high-quality reference translations from the domain):
`{domain_exemplars}`

`{source_lang}` source:
`{source_segment}`

`{target_lang}` translation:
`{target_segment}`

Based on the source segment, machine translation, and domain exemplars surrounded with triple backticks, identify error types in the translation and classify them.

Grounding Rules:
- Terminology Consistency: If the translation uses a term that contradicts a term used in the domain exemplars, flag it as a Terminology error.
- Style Alignment: Use the exemplars to determine the appropriate register (e.g., formal vs. informal) for this domain.
- Evidence: For every identified error, if applicable, reference the specific "`evidence_id`" from the domain exemplars that justifies your diagnosis.

ERROR RUBRIC: Accuracy errors (meaning transfer issues):

- `accuracy/addition`: Translation includes content not present in source (e.g., adding explanatory text)
- `accuracy/mistranslation`: Source meaning incorrectly rendered (e.g., "hot" → "cold", negation flipped)
- accuracy/omission: Source content missing from translation (e.g., clause or word dropped)
- `accuracy/untranslated_text`: Source text left untranslated when translation expected

Fluency errors (target language quality):
- `fluency/grammar`: Grammatical errors (e.g., subject-verb disagreement, wrong tense)
- `fluency/spelling`: Spelling errors or typos
- `fluency/punctuation`: Incorrect punctuation or typography for target locale
- `fluency/character_encoding`: Display issues (e.g., mojibake, garbled characters)
- `fluency/inconsistency`: Same term translated differently without justification
- `fluency/register`: Wrong formality level (e.g., informal "du" in formal document)

Locale convention errors (formatting for target region):
- `locale/currency_format`: Currency not formatted per target locale (e.g., "$100" vs "100 €")
- `locale/date_format`: Date not formatted per target locale (e.g., "12/31/2024" vs "31.12.2024")
- `locale/number_format`: Numbers not formatted per target locale (e.g., "1,000.50" vs "1.000,50")
- `locale/time_format`: Time not formatted per target locale
- `locale/name_format`: Names not ordered per target conventions
- `locale/telephone_format`: Phone numbers not formatted correctly

Terminology errors (domain-specific term issues):
- `terminology/inappropriate_for_context`: Term technically correct but wrong for domain
- `terminology/inconsistent_use`: Domain term differs from exemplars (cite `evidence_id`)

Style errors:
- `style/awkward`: Grammatically correct but unnatural phrasing

Other:
- `other/non_translation`: Output is not a translation (e.g., refusal, error message)
- `other/no_error`: No errors found

SEVERITY LEVELS:
- critical: Errors that inhibit comprehension or could cause harm
- major: Errors that disrupt flow but text is still understandable
- minor: Technical errors that do not disrupt flow or hinder comprehension

CHAIN OF THOUGHT: Before providing the final XML output, think through your evaluation step by step:

1. First, compare source and translation for accuracy (additions, omissions, mistranslations)
2. Then check if exemplars establish terminology patterns relevant to this segment

3. Check fluency issues (grammar, spelling, punctuation)
4. Check locale formatting if applicable
5. Assign severity based on user impact

Requirements:

1. Identify each error with its specific category and subcategory
2. Assign appropriate severity level to each error
3. Provide brief explanation for each identified error
4. Reference `evidence_id` from domain exemplars when applicable
5. If no errors found, respond with "`no-error`"

OUTPUT FORMAT:

Return ONLY the XML structure below. Do NOT include explanations, notes, preambles, or markdown code blocks.

For translations WITH errors (include one `<error>` block per error found):

```
<evaluation>
  <has_errors>true</has_errors>
  <errors>
    <error>
      <category>CATEGORY/SUBCATEGORY</category>
      <severity>critical|major|minor</severity>
      <span>ERRONEOUS_TEXT</span>
      <explanation>BRIEF_EXPLANATION</explanation>
      <evidence_id>EXEMPLAR_ID_IF_APPLICABLE</evidence_id>
    <suggested_edit>CORRECTED_TEXT_FOR_SPAN</suggested_edit>
    </error>
  </errors>
</evaluation>
```

Repeat `<error>` block for each additional error

EDIT CONTRACT:

- `suggested_edit` must be a direct replacement for the text in `<span>`
- For omissions, provide the missing text that should be inserted
- For additions, use empty string or "`[DELETE]`" to indicate removal
- Keep edits minimal - only fix the identified error, preserve surrounding context
- If no clear fix exists (e.g., `style/awkward`), provide the most natural alternative

For translations with NO errors:

```
<evaluation>
  <has_errors>false</has_errors>
  <errors></errors>
</evaluation>
```

E.2 Post-Editor Prompts

Post-Editor Prompt for C_1

You are an expert translation post-editor. Your task is to perform EVIDENCE-TARGETED post-editing, correcting ONLY the specific errors identified by the quality evaluator. Do not make unnecessary changes to parts of the translation that were not flagged.

`{source_lang}` source:
`{source_text}`

`{target_lang}` translation:
`{translated_text}`

EVALUATION RESULTS:

Has Errors: `{has_errors}`

`{errors_section}`

INSTRUCTIONS:

1. **If no errors were identified** (`has_errors: false`):
   - Return the original translation UNCHANGED
   - Do not attempt to "improve" correct translations
2. **If errors were identified** ('`has_errors: true`'):
   - Review each error's category, severity, span, and explanation
   - Make MINIMAL, TARGETED corrections only for the identified errors

- Preserve ALL non-erroneous parts of the translation exactly as they are
- Prioritize critical errors, then major, then minor
- Ensure corrections maintain consistency with the rest of the translation

3. **Correction Guidelines by Error Category**:
   - **Accuracy errors**: Fix mistranslations, add omitted content, remove additions, translate untranslated text
   - **Fluency errors**: Fix grammar, spelling, punctuation while preserving meaning
   - **Terminology errors**: Use suggested, appropriate domain-specific terms based on error detail
   - **Style errors**: Improve awkward phrasing minimally
   - **Locale convention errors**: Adjust formats to target locale standards

IMPORTANT OUTPUT FORMAT:

Return ONLY the XML structure below. Do NOT include explanations, notes, preambles, or markdown code blocks outside the XML.

For translations that NEEDED corrections:

```
<correction>
  <corrected>true</corrected>
  <corrected_translation>FULL_CORRECTED_TRANSLATION_HERE</corrected_translation>
  <changes>
    <change>
      <original_span>ORIGINAL_ERRONEOUS_TEXT</original_span>
       <corrected_span>CORRECTED_TEXT</corrected_span>
     <error_addressed>CATEGORY/SUBCATEGORY</error_addressed>
       <rationale>BRIEF_EXPLANATION_OF_FIX</rationale>
    </change>
  </changes>
</correction>
```

Repeat <change> *block for each correction made*

For translations with NO errors (unchanged):

```
<correction>
  <corrected>false</corrected>
  <corrected_translation>ORIGINAL_TRANSLATION_UNCHANGED</corrected_translation>
  <changes></changes>
</correction>
```

Post-Editor Prompt for C_2

You are an expert translation verifier and post-editor for `{source_lang}` to `{target_lang}` translation.

Your task is to verify whether a previous correction attempt (Post-editor 1) has resolved all identified translation errors, and if not, fix the remaining issues.

INPUT

Source text (`{source_lang}`):

`{source_text}`

Original translation (before correction):

`{original_translation}`

Corrected translation (after Post-editor 1):

`{corrected_translation}`

Original errors identified by evaluator:

`{errors_section}`

Changes made by Post-editor 1:

`{changes_section}`

YOUR TASK

1. **Verify each error**: For each error identified by the evaluator, determine if Post-editor 1's changes have adequately addressed it.
2. **Identify unresolved errors**: List any errors that remain unresolved or were only partially fixed.
3. **Apply additional corrections**: If there are unresolved errors, apply the necessary corrections to fully resolve them.

OUTPUT FORMAT

Return ONLY the XML structure below. Do NOT include explanations, notes, preambles, or markdown code blocks outside the XML.

```
<verification>
  <all_errors_resolved>true/false</all_errors_resolved>
  <error_status>
    <error>
      <original_error>Description of the original error</original_error>
      <status>resolved/partially_resolved/unresolved</status>
      <assessment>Explanation of whether and how it was fixed</assessment>
    </error>
  </error_status>
  <additional_correction_needed>true/false</additional_correction_needed>
  <final_translation>
    The final corrected translation (either the Post-editor 1 output if all resolved,
    or your further corrected version)
  </final_translation>
  <additional_changes>
    <change>
      <original_span>Text that was changed</original_span>
      <corrected_span>New corrected text</corrected_span>
      <error_addressed>Which unresolved error this fixes</error_addressed>
      <rationale>Why this correction is needed</rationale>
    </change>
  </additional_changes>
</verification>
```

Repeat <error> block for each original error

Include only if additional corrections are needed

GUIDELINES

- Be thorough: Check each original error against the corrected translation
- Be conservative: Only make additional changes if truly necessary
- Preserve good corrections: Don't undo successful corrections from Post-editor 1
- Focus on unresolved issues: Don't introduce new changes for issues not in the original error list
- If all errors are resolved, set `all_errors_resolved` to true and return the Post-editor 1 output unchanged

E.3 Prompt for Translation

Prompt for Translation (EN → CN as an example)

You are a professional translator. Your task is to translate the following English text into Chinese. Do not output anything other than a single Chinese translation for the text within <translation> and </translation> XML tag. Return exactly one XML element in the form: <translation> ... </translation>.

The following are {num_exemplars} exemplar translations of similar English texts:

```
<exemplar_1>
  English: {english_1}
  Chinese: {chinese_1}
</exemplar_1>
...
```

This section is only included in the presence of exemplars

Translate the following text:

```
<text>
  {source_text}
</text>
```