AdaptiveMT at WMT26: A Controller–Executor–Reducer Agent Harness for Document-Level Translation

Soyoung Lee
Independent Researcher
sycj427@gmail.com

Abstract

We describe AdaptiveMT, our submission to the English–Korean condition of the WMT26 General Machine Translation task (Kocmi et al., 2026) in the unconstrained track. AdaptiveMT is a language-general adaptive translation harness with target-language-specific profiles; this run uses the Korean profile. The system uses a controller–executor–reducer agent harness. For each document, the harness analyzes the source, generates independent candidate translations from two heterogeneous models, compares them with token-level and semantic diffs, and synthesizes a final translation with a judge model. Mandatory QA hooks and deterministic Korean-specific guards run before finalization; failures trigger bounded local repair. Multimodal documents use structured visual context extracted from images or sampled video frames before translation. We processed all 2,078 distributed blindset records under two Korean target tags; the subsequently released General MT evaluation set contains 198 English-to-Korean documents tagged `kor_Hang`. On this official subset, the harness made 1,441 successful LLM calls at an estimated list-price inference cost of $12.03, and all 198 documents finished with passing hooks. In blindset-wide diagnostics, synthesis eliminated observed candidate-level mechanical errors, and a GPT-5 judge preferred the final output over a bare GPT-4.1 baseline on 600 sampled documents (44.5% vs. 30.0%, 25.5% ties).

1 Introduction

Recent WMT General MT evaluations (Kocmi et al., 2024) show that strong LLMs translate well when prompted directly, yet single-pass prompting still leaves document-level failure modes: dropped or merged lines, untranslated source fragments, broken placeholders and markup, register mismatches in dialogue, and instruction violations on samples with explicit constraints. These errors are individually rare but recur across a large blindset, and many are easier to expose when an independent model is asked to compare or verify the output.

Our WMT26 submission, **AdaptiveMT**,[1] therefore treats translation as a bounded agent harness rather than a single call. The design borrows three ideas from recent agent harnesses for software engineering: (i) a typed state object that all steps read and write, (ii) a fixed action vocabulary whose preconditions are enforced by a normalizer rather than trusted to the model, and (iii) forced verification hooks that stand between the agent and the FINISH action. Within this frame, translation quality comes from *post-decoding soft fusion*: two heterogeneous candidate models translate independently, their outputs are compared by token-level and semantic diffs, and a judge model synthesizes the final translation from diff-localized evidence, with the option to reject both candidates and translate directly from the source where both are awkward.

We participated in the English→Korean condition only. At submission time, we processed all 2,078 records carrying either of the two Korean target tags in the distributed blindset. The subsequently released General MT evaluation subset contains 198 `kor_Hang` documents, including 174 with image or video context. Section 2 positions the design against prior work, Section 3 describes the harness, Section 4 the Korean target-language profile, Section 5 the submission run, and Section 6 what we learned.

2 Related Work

LLMs for (document-level) MT. General-purpose LLMs are now competitive with or better than dedicated MT systems for many high-resource pairs (Hendy et al., 2023; Vilar et al., 2023), and recent WMT General MT findings confirm that LLM-based systems dominate the top ranks while document-level and instruction-

[1] Code: https://github.com/8pril/adaptivemt

Proceedings of the Eleventh Conference on Machine Translation, pages 1318–1326
October 28-29, 2026 ©2026 Association for Computational Linguistics

following errors persist (Kocmi et al., 2023, 2024). Wang et al. (2023) show that prompting LLMs with full documents improves discourse phenomena over sentence-level translation; our harness also translates whole documents, but adds structural analysis and hook-based verification.

LLM-as-a-judge and quality estimation. LLMs are strong reference-free evaluators of translation quality (Kocmi and Federmann, 2023) and are widely used as pairwise or scalar judges (Zheng et al., 2023). We use a judge model not to score candidates but to *synthesize* from them, conditioned on explicit token-level and semantic diff evidence; our QA hooks are a second, targeted use of LLM judging with a deterministic guard layer that can overrule the judge.

Self-refinement and agentic inference. Iterative self-correction (Madaan et al., 2023; Shinn et al., 2023) and tool-using agent loops (Yao et al., 2023) improve LLM output quality at inference time, and structured agent–computer interfaces are central to software-engineering agents such as SWE-agent (Yang et al., 2024). Translation-specific multi-step strategies include MAPS (He et al., 2024), which elicits translation-relevant knowledge before translating, and multi-agent pipelines for literary translation (Wu et al., 2024). Our harness differs in that the loop is *bounded and normalized*: the action vocabulary is fixed, action preconditions are enforced in code, revision is budgeted, and finalization is guarded by forced hooks rather than left to the agent's judgment.

System combination. Combining multiple MT systems predates LLMs, from confusion-network combination (Rosti et al., 2007) to MBR and quality-aware decoding with neural metrics (Freitag et al., 2022; Fernandes et al., 2022). Our post-decoding soft fusion is a text-level analogue: instead of ensembling token probabilities or selecting a whole candidate, a judge model composes the final translation from diff-localized evidence over two heterogeneous candidates and may retranslate parts where both fail.

3 System Architecture

3.1 Harness loop

Each document sample is wrapped in an `AgentState` containing the source document, the sample-level instruction, sample-supplied format constraints, deterministically extracted format-safety constraints, and accumulating artifacts (multimodal context, structured source analysis, candidates, diffs, the final translation, repair patches, hook results, a full event log, and LLM-call logs). The harness then loops:

```
while not finished and budget remains:
action = select(state)
obs = execute(action, state)
state = reduce(state, action, obs)
force_hooks_if_needed(state)
```

Figure 1 shows the per-document execution flow of the submitted configuration. Its core action sequence uses EXTRACTMULTIMODALCONTEXT, ANALYZESOURCE, GENERATECANDIDATE, DIFFCANDIDATES, SYNTHESIZEFINAL, CHECKALL (a combined run of four QA hooks), REVISEFINAL, and FINISH. The fixed action vocabulary also contains optional research, terminology, and fact-checking actions. When enabled, these actions retrieve external evidence, construct an evidence-backed terminology table, and extract factual constraints before candidate generation; their artifacts are stored in `AgentState` and made available to downstream translation steps. All were disabled for the submission and are omitted from the figure. A *normalizer* validates every proposed action against the state: FINISH is rejected until a final translation exists and all required hooks have run, SYNTHESIZEFINAL is rejected before at least two candidates and a diff exist, and disabled or repeated actions fall back to a deterministic next step.

The action selector supports three modes. `always` implements a *bounded agent*: the LLM controller selects every next action from the fixed vocabulary, subject to the normalizer and mandatory finalization hooks. `minimal` is partially agentic, consulting the controller only for ambiguous repair decisions after a hook failure. `never` implements a deterministic agentic workflow in which the state-based `fallback_action()` policy selects every action. All three modes share the same executor, reducer, normalizer, and forced hooks. The submission used `never` for repeatability and cost control, and is therefore a *deterministic agentic workflow* rather than an autonomous agent.

3.2 Source analysis

ANALYZESOURCE asks an analysis model to convert the sample into structured JSON: explicit constraints (hard rules, format, style, preserve/omit

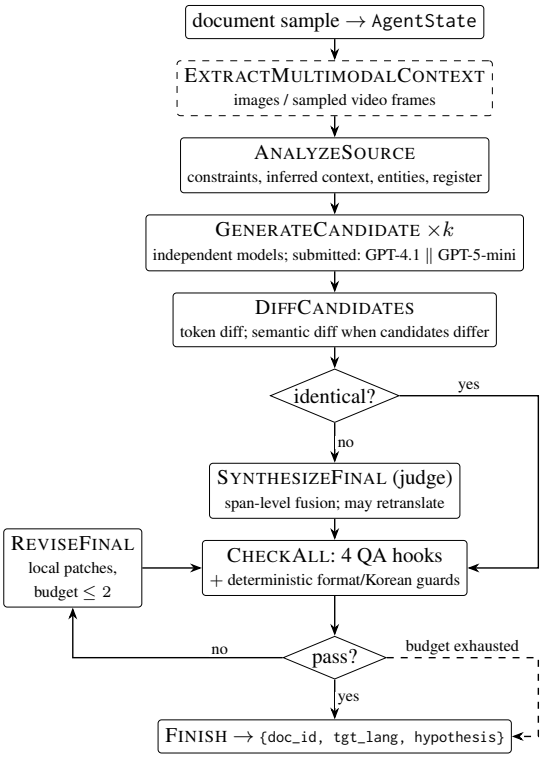

Figure 1: The submission-time AdaptiveMT flow for one document. Solid boxes are actions of the deterministic policy used for the submission; the dashed box runs only for documents with multimodal input. When the two cleaned candidates are byte-identical, judge synthesis is skipped and the shared translation goes directly to validation. Failed hooks trigger at most two local-repair passes; when the repair budget is exhausted, the document finishes best-effort (dashed edge).

lists) derived from the sample instruction and format constraints; an *inferred context* with likely domain and genre, a confidence level, and supporting evidence; a discourse summary and outline; an entity map; coreference notes; and tone/register. Explicit sample instructions always take precedence over inferred context. This avoids relying on hand-written genre heuristics, which would not generalize across the blindset's mix of literary prose, social-media posts, subtitles, and technical strings.

3.3 Candidate generation and diff-based synthesis

Two candidate models translate the full document independently, conditioned on the source analysis, format-safety constraints, derived style requirements, and any multimodal context. When the cleaned candidates differ, DIFFCANDIDATES produces two artifacts:

- a **localized token-level diff**, obtained by deterministically aligning the cleaned outputs with a custom tokenizer over markup tags, printf placeholders, Hangul spans, Latin tokens, whitespace, and symbols, then grouping adjacent token insertions, deletions, and replacements with surrounding context; and
- a **structured semantic diff** produced by the judge from the source, instructions, contextual artifacts, both candidates, and the token diff. It records meaning and coverage divergences, terminology, fact, style, and format risks, together with a preferred rendering and rationale for each disputed span.

SYNTHESIZEFINAL gives the judge model both candidates and both diffs and asks for a single final translation. The judge may take diff-localized portions from either candidate or ignore both and translate a part directly from the source when both candidates are wrong. This is post-decoding, text-level soft fusion: no token probabilities are ensembled at decoding time. When the two cleaned candidates are byte-identical, the harness uses the shared translation as its initial final output and skips the semantic diff and the synthesis call. Across the full 2,078-record processing run, this shortcut fired on 234 documents (11.3%), concentrated in short dialogue lines. Four of these translations were subsequently changed by the hook-and-repair stage. None of the 198 documents in the official evaluation subset had identical candidates.

Figure 2 illustrates the span-level fusion process with an example from the full processing run.

3.4 Validation hooks and bounded repair

Before the harness may finish, four hooks must run: *instruction following*, *glossary compliance*, *format preservation*, and *completeness* (no dropped, merged, or hallucinated lines). The default CHECKALL action evaluates all four in one checker-model call. Hook results are then post-processed by deterministic guards (Section 4) that can flip a hook to failed regardless of the LLM's verdict.

On failure, REVISEFINAL proposes *local repair patches*, which are span replacements targeted at the reported issues. Each proposed span replacement is applied once; patches that cannot be matched exactly are delegated to a constrained patch-applier model. The revised translation is then checked again. Repair is budgeted (up to two revisions per document in the submission run); when

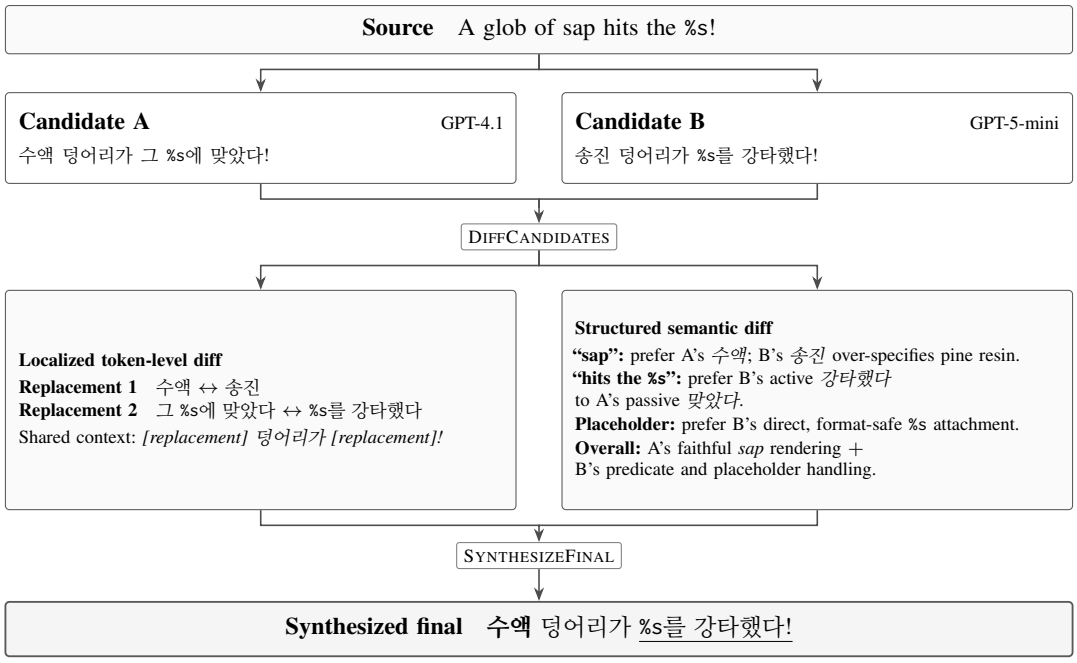

Figure 2: Span-level synthesis, abridged and reformatted from a recorded trace. The diffs favor A's faithful rendering of *sap* and B's predicate and placeholder handling. The resulting translation matches neither candidate and was accepted without repair (bold = from A; underlined = from B).

the budget is exhausted the harness finishes best-effort rather than looping. Across the full processing run, 45 documents (2.2%) went through repair: 42 with one revision and 3 with two. Within the official 198-document subset, three documents received one revision each and all converged to passing hooks. The sole best-effort finish with a failing hook occurred outside that subset.

3.5 Multimodal context extraction

The official 198-document evaluation subset includes 174 documents carrying an image (social-media screenshots) or a video file (dialogue clips). For these, EXTRACTMULTIMODALCONTEXT runs before any translation step: images are sent directly to a vision model; videos are sampled into at most six frames with `ffmpeg`, with the frame count scaled to clip duration. The extractor returns a *visual-only* structured context, including scene, participants, on-screen text, and register cues, which is injected into candidate, synthesis, hook, and repair prompts. The extractor is explicitly forbidden from inventing translation rules; it only reports what is visible, and the translation prompts use it to resolve deixis, speaker relationships, tone, and ambiguous referents. Media context also drives derived style requirements: video-sourced samples are treated as spoken dialogue and steered toward natural colloquial Korean endings (-야/-어/-지/-네) rather than literary register (-다/-습니다), unless the speaker's register indicates otherwise. In the submitted implementation, translation prompts carry the sample's main instruction verbatim; the separate multimodal-instruction field is used only by the keyword heuristic that derives style requirements. This was sufficient for the WMT26 blindset: across all 174 multimodal documents, the two instruction fields are near-duplicates, and the only unique content in the multimodal instruction is the video-speech cue that the heuristic converts into register requirements. For future multilingual and multimodal settings, we would pass both fields verbatim.

4 Target-Language Profile: Korean

The harness loop, action set, hooks, and synthesis logic of Section 3 are language-agnostic; everything specific to a target language, including deterministic guards, register steering, and script checks, is isolated in a *target-language profile*. Supporting a new language pair means writing a new profile, not a new harness. This mirrors the separation that software-engineering agent harnesses draw between a generic agent loop and pluggable, domain-specific extensions. The submitted system instantiates the Korean profile.

LLM-based hooks occasionally miss mechanical failures, so the profile layers cheap deterministic

checks on top:

Format safety. Printf placeholders (%s, %d, ...) are extracted from the source before translation and must be preserved exactly, in count and order. Notation and separator symbols between word characters (^, |, ->, ...) are likewise protected unless the sample instruction says to rewrite them.

Hangul-ratio guard. For Korean targets, a translation of a non-trivial source that contains almost no Hangul, or vastly more Latin than Hangul text, is flagged as untranslated. A near-identity check (normalized similarity > 0.82 against the source) catches copy-through failures.

Untranslated-Latin-token guard. Latin source tokens that survive verbatim into the Korean output are flagged unless they are format-safe (URLs, handles, hashtags, markup, acronyms, code-like tokens) or the sample instruction explicitly licenses preservation. In the submitted system this exemption applied at the sample level: any sample licensing foreign-word preservation was skipped wholesale (later refined; see Section 6). Such an instruction occurs in 133 official-set documents. Flagged tokens are appended to the instruction-following hook as failures, forcing a repair pass that must translate, transliterate, or justify each token.

These guards convert the most common silent LLM failure modes, including copy-through, partial translation, and placeholder corruption, into visible hook failures with a targeted repair path.

5 Submission

Evaluation scope. The mixed General MT/test-suite blindset contained 1,880 ko_KR and 198 kor_Hang records.[2] Because the official English–Korean tag was unspecified, we processed both; the subsequent release identifies kor_Hang as the 198-document evaluation subset. We report it separately from the 2,078-record full run.

Configuration. We submitted to the unconstrained English→Korean condition. The run used deterministic control (never mode), two candidates (GPT-4.1 and GPT-5-mini), GPT-4.1 for semantic diffing, synthesis, and repair patching, Upstage Solar-Pro3 for source analysis and combined QA, and GPT-4.1 vision for multimodal extraction, with at most 12 loop steps and up to two revisions per document.

For reproducibility, GPT-4.1 calls left temperature and top_p unset, using the Responses API defaults of 1.0 for both. During batched candidate generation, GPT-5-mini's reasoning effort was explicitly set to low after early batches had used the provider default (medium at the time).[3] The OpenAI API calls used the aliases gpt-4.1 and gpt-5-mini. For reference, OpenAI lists the dated versions gpt-4.1-2025-04-14 and gpt-5-mini-2025-08-07, respectively.[4] Upstage reports Solar-Pro3 as a mixture-of-experts model with 102B total parameters and 12B active parameters per token.[5] OpenAI does not disclose parameter counts for GPT-4.1 or GPT-5-mini.

Run statistics. Table 1 separates the official evaluation subset from the broader processing run. Every processed record received exactly one output row with a non-empty hypothesis. The harness logs a full per-document trace (every action event, LLM call, and intermediate artifact); statistics retain the last trace record per document after resumed runs, which is the source of every statistic in this section and supports the error analysis in Section 6.

Results. WMT26 did not release preliminary automatic scores. At camera-ready preparation time, finalized official human rankings were unavailable, so we report no official shared-task results. The task and its human evaluation are described in the shared-task findings paper (Kocmi et al., 2026).

Blindset-wide stage audit. The released Korean documents contain no reference translations, so a reference-based quality ablation is not possible. To isolate the contributions of synthesis and repair, we instead reconstructed intermediate outputs from the trace and applied three deterministic checks with unambiguous pass/fail semantics: printf-placeholder preservation, structure alignment under the official checker criterion, and non-empty output. Table 2 compares the bare baseline, both independent candidates, the first post-synthesis translation before repair, and the final full-

[2] Organizer clarification: https://groups.google.com/g/wmt-tasks/c/zLmAgJGU2UU. Released task data: https://github.com/wmt-conference/wmt26-general-mt.

[3] Low effort was used for 142/215 official-subset and 1,686/2,113 full-run GPT-5-mini candidate calls; the remainder used the then-default medium effort.

[4] OpenAI model pages: https://developers.openai.com/api/docs/models/gpt-4.1 and https://developers.openai.com/api/docs/models/gpt-5-mini.

[5] https://www.upstage.ai/blog/en/solar-pro-3-0323

	Official	Full run
Documents	198	2,078
multimodal	174	174
Successful LLM calls	1,441	12,828
per document	7.3	6.2
Latency (s), median	68.2	26.4
P90	145.5	53.4
Tokens, input	3.90M	15.6M
output	1.32M	5.7M
Cost (USD, est.), harness	$12.03	$46.78
per document	$0.061	$0.023
bare baseline	$0.61	$1.17
harness/baseline	19.7×	40.1×
Identical shortcuts	0	234 (11.3%)
Documents repaired	3 (1.5%)	45 (2.2%)
Failing-hook finishes	0	1

Table 1: Run statistics. API-attempt counts including retries were 1,616 (official) and 13,003 (full). Costs use September 2026 list prices; harness usage was recorded and baseline content-token counts were reconstructed (Section 6). Latency sums sequential action durations. The sole failing hook was a guard false positive outside the official subset.

Stage	Placeh.	Struct.	Empty	Hook fail.
Bare GPT-4.1	5	0	0	–
GPT-4.1 candidate	6	0	1	–
GPT-5-mini candidate	2	4	1	–
Post-synthesis	**0**	**0**	**0**	45
Final output	**0**	**0**	**0**	1

Table 2: Stage-wise diagnostic over all 2,078 processed records. The first three numeric columns count mechanical output errors; counts may overlap. Structure mismatch follows the official alignment-checker criterion (top-level element counts for JSON documents and <p> counts for HTML documents; other document types are not covered). Hook failures are defined only after synthesis and repair.

run output. It also reports the number of records with at least one failing mandatory hook at the two stages where hooks are defined. The audit covers all 2,078 records in the broader processing run rather than only the 198 official documents.

All candidate-level mechanical errors disappear immediately after synthesis. The two models did not break the same placeholder in the same record, so the judge selected compliant renderings from the diff evidence. Synthesis also recovered from one empty candidate from each model and prevented GPT-5-mini's four JSON/HTML mismatches, one of which overlaps with its empty candidate, from propagating. Repair reduced records with failing hooks from 45 to one (and from three to zero on the official subset). Thus the audit attributes the observed mechanical gains to synthesis, while repair's measured contribution is the resolution of hook failures in 44 records. Because the hooks are part of the system rather than independent quality judgments, neither result by itself establishes overall translation quality.

What synthesis does. Because two-candidate provenance is undefined when either candidate is empty, the two records described above are excluded from this analysis. Comparing the remaining 2,076 records against the stored candidates shows the judge does not simply default to one model: the final translation equals both stored candidates in 230 cases (11.1%), only the GPT-4.1 candidate in 516 (24.9%), only the GPT-5-mini candidate in 979 (47.2%), is a span-level mix or light edit of the closer candidate in 339 (16.3%; SequenceMatcher ratio ≥ 0.6, median 0.93 among these records), and is substantially rewritten in the remaining 12 (0.6%). The first count is four lower than the 234 identical-candidate shortcuts in Table 1 because four shared translations were changed during subsequent hook-driven repair.

Blindset-wide pairwise quality comparison. To add a quality signal on top of the mechanical audit, we ran a pairwise LLM-judge comparison between the bare GPT-4.1 baseline and the full-run output on a random sample of 600 of the 1,871 records where the two systems produced different strings (sampled with a fixed-seed script). The judge used the `gpt-5` API alias with low reasoning effort; OpenAI lists its corresponding snapshot as `gpt-5-2025-08-07`.[6] GPT-5 was not used anywhere in the submitted pipeline (the submission judge was GPT-4.1). It saw the sample's full instruction (including the multimodal instruction where present), the source, and both translations. Position bias is controlled by judging each document twice with the order swapped; a system wins only if it wins both orders, otherwise the document is a tie. The full-run output was preferred on 267 documents (44.5%), the bare baseline on 180 (30.0%), with 153 ties (25.5%); a two-sided sign test over the 447 decisive pairs gives $p < 10^{-4}$. As with the mechanical audit, this comparison covers the broader processed blindset rather than only the subsequently identified official subset. Inspecting the baseline's wins shows they are dominated by fine-grained stylistic preferences on short game

[6] https://developers.openai.com/api/docs/models/gpt-5

strings rather than systematic defects in the full-run output. Latin retention does not explain the losses in aggregate: guard-flag rates are essentially equal in the two win groups, and documents with retained proper nouns concentrate in ties. However, Latin retention is the salient difference in 10 of the 180 losses, where the full-run output kept source-language names or phrases that the baseline transliterated or translated; this is consistent with the sample-level exemption in effect at submission time, later replaced by the token-level guard (Section 6).

6 Discussion

Strengths. The diff-based synthesis step is most valuable when the two candidates disagree meaningfully: disagreement localizes exactly the spans where at least one model is wrong, and the judge adjudicates with both renderings visible. Conversely, in the full processing run, 234 records (11.3%) had byte-identical candidates and needed no judge, although four were subsequently modified by hook-driven repair and the shortcut did not occur in the official 198-document subset. Heterogeneous-model agreement is a cheap confidence signal that we exploit to skip work.

Failure modes. In the broader processing run, one `ko_KR` record outside the official subset finished with a failing hook that turned out to be a guard false positive: the source described a military launcher whose model designation *M202A1* was correctly preserved in the Korean output, but the untranslated-Latin-token guard's format-safe patterns did not cover alphanumeric designations with alternating letters and digits, and the repair model correctly kept the designation unchanged. The output was fine; the guard was wrong. More generally, under a bounded revision budget, a disagreement between a deterministic guard and the repair model cannot be resolved within the loop and surfaces as a best-effort finish.

Post-submission guard changes. Error analysis after submission led to two guard changes. First, the format-safe pattern was widened to cover alphanumeric model designations such as *M202A1*, removing the false positive above. Second, we found the guard too permissive on samples whose instructions allow foreign words to be kept unchanged: the exemption applied at the sample level, so transcript interjections and sound-effect-like tokens (e.g., *MM*, *Oh*) could also remain untranslated. The guard is now token-level: exact preservation of proper names and foreign words is honored only where the sample instruction licenses it, while interjection-like Latin tokens are still checked by the guard. The preservation check also reads the multimodal instruction in addition to the main instruction. These changes postdate the submission: the outputs and every number reported in this paper reflect the pre-change run.

Cost. We estimated inference cost from the uncached-input, cached-input, and output-token usage recorded for every successful LLM call in the trace. At the providers' published per-million-token list prices, the official 198-document subset cost \$12.03 (\$0.061 per document) and the full 2,078-record processing run cost \$46.78 (\$0.023 per record).[7] The estimate uses token usage from successful responses; failed retry attempts reported no usage. It excludes the post-hoc bare baseline and pairwise evaluation, taxes, account credits, and volume or batch discounts. For comparison, we reconstructed the post-hoc bare GPT-4.1 baseline's cost from its exact prompts and saved outputs because its runner did not retain API usage. Applying the uncached GPT-4.1 rates gives approximately \$0.61 (\$0.0031 per document) on the official subset and \$1.17 (\$0.00056 per record) on the full run. These estimates assume one successful call per record and exclude message-framing tokens and any unrecorded retries. The harness therefore cost approximately $19.7\times$ the bare baseline on the official subset and $40.1\times$ over the full run. The larger full-run ratio reflects the 1,880 additional, predominantly short non-multimodal records, for which a single-call baseline is especially inexpensive.

Latency. We measured end-to-end per-document latency as the sum of the sequential action-event durations retained in the trace. Median and P90 latency were 68.2 and 145.5 seconds on the official subset, and 26.4 and 53.4 seconds over the full processing run.[8] The bare baseline runner did not

[7] Per-model costs (official/full run): \$10.94/\$41.95 for GPT-4.1, \$0.85/\$3.56 for GPT-5-mini, and \$0.24/\$1.26 for Solar-Pro3. Uncached-input/cached-input/output rates: \$2.00/\$0.50/\$8.00 for GPT-4.1, \$0.25/\$0.025/\$2.00 for GPT-5-mini, and \$0.15/\$0.015/\$0.60 for Solar-Pro3. Published pricing, accessed September 2026. GPT-4.1: `https://openai.com/index/gpt-4-1/`. GPT-5-mini: `https://developers.openai.com/api/docs/models/gpt-5-mini`. Solar-Pro3: `https://www.upstage.ai/pricing/api`.

[8] We compute P90 by the nearest-rank method: after sorting n observations, select the observation at rank $\lceil 0.9n \rceil$.

retain timing data, so we cannot report a directly observed baseline increase. As a same-run single-call proxy, the GPT-4.1 candidate-generation call had median latency of 7.80 seconds on the official subset and 2.76 seconds over the full run. The median per-document ratio of harness latency to this proxy was $8.54\times$ and $8.93\times$, respectively. These are deployment-level observations that include network latency, provider load, retries, and local multimodal preprocessing; they do not isolate model inference time and should not be interpreted as hardware-independent speed measurements. The submitted harness executed actions sequentially, so parallel candidate generation could reduce latency without changing the pipeline logic.

7 Conclusion

We described AdaptiveMT, a stateful controller–executor–reducer harness for the WMT26 General MT English–Korean condition. The system combines document-level candidate generation, diff-based synthesis over heterogeneous models, normalized action selection, forced QA hooks, and Korean-specific deterministic guards. On the official 198-document evaluation subset, the system made 1,441 successful LLM calls, repaired three documents, and finished every document with passing hooks. In our broader blindset-wide diagnostic, synthesis eliminated the mechanical output errors present in the single-model outputs, and repair resolved the hook failures in 44 of 45 affected records (Table 2). Finally, swappable target-language profiles separate Korean-specific behavior from the core harness.

Limitations

While the harness itself is language-agnostic and language-specific behavior is isolated in a target-language profile (Section 4), only the Korean profile has been implemented and tuned; the portability claim is a design property that has not yet been demonstrated empirically on another language pair. All component models are closed commercial LLMs accessed through APIs; stochastic decoding means that reruns may not reproduce identical outputs. We report no reference-based quality ablation because the released Korean documents contained no reference translations. The blindset-wide stage audit (Section 5) isolates synthesis and repair only through mechanical failure proxies and internal hook outcomes. That audit and the pairwise comparison cover the broader 2,078-record processing run rather than a separately resampled official subset; the latter also relies on an LLM judge that shares a provider family with the candidate models and cannot see the multimodal inputs. Without finalized official human results at camera-ready time, the reported comparisons should be read as post-hoc diagnostics rather than official shared-task results or a complete evaluation of translation quality.

References

Patrick Fernandes, António Farinhas, Ricardo Rei, José G. C. de Souza, Perez Ogayo, Graham Neubig, and Andre Martins. 2022. Quality-aware decoding for neural machine translation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1396–1412, Seattle, United States. Association for Computational Linguistics.

Markus Freitag, David Grangier, Qijun Tan, and Bowen Liang. 2022. High quality rather than high model probability: Minimum Bayes risk decoding with neural metrics. *Transactions of the Association for Computational Linguistics*, 10:811–825.

Zhiwei He, Tian Liang, Wenxiang Jiao, Zhuosheng Zhang, Yujiu Yang, Rui Wang, Zhaopeng Tu, Shuming Shi, and Xing Wang. 2024. Exploring human-like translation strategy with large language models. *Transactions of the Association for Computational Linguistics*, 12:229–246.

Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are GPT models at machine translation? a comprehensive evaluation. *arXiv preprint arXiv:2302.09210*.

Tom Kocmi, Ekaterina Artemova, Eleftherios Avramidis, Rachel Bawden, Ondřej Bojar, Sergey Dukanov, Anton Dvorkovich, Mark Fishel, Markus Freitag, Samuel Frontull, Thamme Gowda, Roman Grundkiewicz, Barry Haddow, Stan Kharevich, Philipp Koehn, Zheng Li, Jean Maillard, Christof Monz, Alexander Murauski, and 12 others. 2026. Findings of the WMT26 general machine translation shared task: Contrastive dynamic human evaluation at scale. In *Proceedings of the Eleventh Conference on Machine Translation*. Association for Computational Linguistics.

Tom Kocmi, Eleftherios Avramidis, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Markus Freitag, Thamme Gowda, Roman Grundkiewicz, Barry Haddow, Marzena Karpinska, Philipp Koehn, Benjamin Marie, Christof Monz, Kenton Murray, Masaaki Nagata, Martin

Popel, Maja Popović, and 3 others. 2024. Findings of the WMT24 general machine translation shared task: The LLM era is here but MT is not solved yet. In *Proceedings of the Ninth Conference on Machine Translation*, pages 1–46, Miami, Florida, USA. Association for Computational Linguistics.

Tom Kocmi, Eleftherios Avramidis, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Markus Freitag, Thamme Gowda, Roman Grundkiewicz, Barry Haddow, Philipp Koehn, Benjamin Marie, Christof Monz, Makoto Morishita, Kenton Murray, Masaaki Nagata, Toshiaki Nakazawa, Martin Popel, and 3 others. 2023. Findings of the 2023 conference on machine translation (WMT23): LLMs are here but not quite there yet. In *Proceedings of the Eighth Conference on Machine Translation*, pages 1–42, Singapore. Association for Computational Linguistics.

Tom Kocmi and Christian Federmann. 2023. Large language models are state-of-the-art evaluators of translation quality. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 193–203, Tampere, Finland. European Association for Machine Translation.

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. In *Advances in Neural Information Processing Systems 36*.

Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie Dorr. 2007. Combining outputs from multiple machine translation systems. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 228–235, Rochester, New York. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36*.

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, Toronto, Canada. Association for Computational Linguistics.

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.

Minghao Wu, Jiahao Xu, Yulin Yuan, Gholamreza Haffari, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024. (perhaps) beyond human translation: Harnessing multi-agent collaboration for translating ultra-long literary texts. *arXiv preprint arXiv:2405.11804*.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems 37*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

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. In *Advances in Neural Information Processing Systems 36 (Datasets and Benchmarks Track)*.