Language Twin: A Shared-State Architecture for Terminology-Consistent Document Translation with Human Edit Propagation—A Pilot Study

Elliott Ahn (Seok-hyun) *

Posted Date: 30 March 2026

doi: 10.20944/preprints202603.2397.v1

Keywords: document translation; shared state; terminology management; human-in-the-loop; post-edit propagation; translation memory; lazy context loading; large language models

 doi:10.20944/preprints202603.2397.v1

Article

Language Twin: A Shared-State Architecture for Terminology-Consistent Document Translation with Human Edit Propagation—A Pilot Study

Elliott Ahn (Seok-Hyun)

Adjunct Professor, Gwangju Institute of Science and Technology (GIST), Republic of Korea; orcid.org/0009-0005-4666-5512; seokhyun.ahn@gmail.com or elliott.ahn@gist.ac.kr

Abstract

We propose Language Twin, a shared-state architecture that organizes translation projects as seven versioned layers (L0–L6), supporting selective context loading, scoped human-edit propagation, and reversible updates. A pilot study translated three curated English-to-Korean document bundles (17 segments) using GPT-4o with temperature 0.3. The Language Twin condition (P1) achieved numerically higher preferred-term accuracy than the strongest baseline (17/21 vs. 14/21; not statistically significant at this sample size) and observed no repeated downstream errors in the monitored set (0/5 vs. 5/5 against the propagation-disabled ablation; Fisher's exact $p = 0.008$), while reducing prompt tokens by 39.2% relative to full-context loading (A4). In blinded human evaluation (quadratic-weighted $\kappa = 0.71$–0.78), P1 achieved the highest terminology rating (4.38/5 vs. 3.97/5) and lowest post-editing time (16.9 s vs. 19.1 s per segment). These pilot-scale results indicate that governed shared state can improve terminology consistency and editing efficiency.

Keywords: document translation; shared state; terminology management; human-in-the-loop; post-edit propagation; translation memory; lazy context loading; large language models

1. Introduction

Recent advances in neural machine translation (NMT) and large language models (LLMs) have substantially improved sentence-level translation quality across many language pairs and domains [1,2]. However, high-stakes document translation in medicine, law, finance, and technical engineering still poses challenges not solved by local generation quality alone. In such settings, translation quality depends on whether the system can maintain terminology consistency across an entire document, preserve discourse coherence across paragraphs and sections, and accumulate corrections as reusable project knowledge rather than isolated edits.

In practice, professional translation workflows combine source documents, glossaries, translation memories, style guides, model prompts, review comments, and quality reports. These assets are usually distributed across disconnected files and tools. As a result, a terminology decision made by a reviewer may not be inherited by later segments, an approved phrasing choice may remain buried in a comment thread, and a correction made in one document may have to be rediscovered in the next. This fragmentation produces repeated errors, duplicated post-editing effort, weak auditability, and poor reproducibility.

Several research strands address important parts of this problem. Document-level MT models improve access to wider context [3–7]; terminology constraint methods bias or enforce approved lexical choices [8–12]; post-editing and interactive MT studies show how human feedback can improve quality [13–16]; translation-memory or retrieval-augmented prompting methods reuse prior examples during generation [17–21]; and interactive CAT environments have explored continuous integration of post-edits and productivity measurement [27,30,31]. Yet these lines of work usually

 doi:10.20944/preprints202603.2397.v1

treat context as an ephemeral input to a model invocation rather than as a persistent project state with provenance, update rules, and rollback semantics.

We argue that the core bottleneck is therefore architectural. The problem is not only that individual models make errors; it is that modern document translation systems lack an explicit operational representation of what has been learned, approved, corrected, and evaluated so far. What must persist is not only source context, but also approved terminology, discourse cues, audience-specific variants, human corrections, and quality signals. Once this state becomes explicit, the system can retrieve only what is relevant at each step, propagate approved edits immediately, and expose a transparent history of how translation decisions evolved.

To address this gap, we propose Language Twin, a shared-state architecture for document translation that organizes project knowledge as layered, non-destructive overlays on a common language state. A central hub maintains versions, scopes, and synchronization across documents; a lazy retrieval module loads only the layers relevant to the current segment; and a human-in-the-loop update mechanism writes accepted edits back into the shared state so that future segments inherit them. The name "Language Twin" draws on the digital twin concept from systems engineering [32,33], in which a continuously synchronized digital model mirrors a physical system's state and supports monitoring, prediction, and control. We adopt the layered-state and continuous-synchronization principles rather than the full-fidelity simulation semantics of industrial digital twins; the system is a workflow substrate, not a physics simulation.

The contributions of this study are fourfold: (1) we formalize a layered shared-state representation for document translation; (2) we define a central hub and lazy loading policy that separate persistent project memory from transient prompting; (3) we demonstrate, in a pilot study on three English-to-Korean document bundles, how governed edit propagation and lazy loading affect lexical control, recurrence, prompt cost, and post-edit effort; and (4) we complement automatic metrics with a blinded human evaluation and a reproducible submission archive containing executed tables, leakage controls, and evaluation artifacts.

2. Related Work

2.1. Document-Level Translation and Discourse Coherence

Document-level MT has long targeted phenomena that sentence-isolated systems handle poorly, including discourse continuity, lexical cohesion, and cross-sentence reference. The field has matured enough to support broad surveys of modeling and evaluation strategies [3]. Memory-based and context-aware decoders incorporate source-side and target-side document context into translation decisions [4,5], while lexical-consistency methods explicitly encourage stable translation of repeated expressions across a discourse [6]. More recently, LLM-era frameworks such as SubDocTrans inject topic-level and local knowledge to improve coherence and reduce omissions in document translation [7], and work on human-like translation strategies with LLMs has shown that deliberate strategy planning can improve both adequacy and naturalness in longer translations [27]. These works are directly relevant, but they focus primarily on generation-time conditioning. They do not define how terminology decisions, reviewer feedback, or project-specific corrections are versioned, governed, and reused across an ongoing translation workflow.

2.2. Terminology Control in Machine Translation

Terminology control has been studied through training-time annotations, constrained decoding, finite-state approaches, robustness analyses, and dictionary-based prompting [8–12]. These methods are highly relevant in domains where incorrect term choices have regulatory or safety implications. However, terminology is often treated as a static constraint list provided to a single decoding run. Much less attention is paid to provenance, conflict resolution, update timing, or selective activation of only those terminology entries relevant to the current segment. In Language Twin, terminology is

 doi:10.20944/preprints202603.2397.v1

not merely injected into a prompt or decoder; it is modeled as a persistent layer with scope, priority, and revision history.

2.3. Human-in-the-Loop Translation and Post-Editing

Human feedback remains indispensable in high-value translation workflows. Continuous learning from post-edits has shown that corrections can improve MT over time [13], while recent LLM work has explored error-annotation-guided post-editing [14], contextual refinement at sentence and document level [15], and iterative self-correction [16]. Interactive CAT environments have further demonstrated that tighter integration of human corrections with system state can yield measurable productivity gains [30], and research on MT quality and post-editing effort has clarified how output quality affects the time and cognitive load required from human editors [31]. These studies demonstrate that post-editing can be more than a terminal clean-up phase. Nevertheless, the accepted edit is still too often treated as the end of the pipeline. The central idea of the present work is that an accepted human edit should become reusable shared state that can constrain or guide the remainder of the current document and, where appropriate, future documents in the same project.

Recent shared-task results in automatic post-editing also suggest caution: even when baseline translations are already strong, generic APE does not guarantee consistent improvements across settings [29]. This motivates a narrower operational target. Rather than promising universal rewriting gains, Language Twin is designed to reduce recurrence of already observed error types by converting approved corrections into scoped, reusable state.

2.4. Translation Memory, CAT, and LLM-Augmented Translation

Translation memory (TM) and CAT environments have long supported segment reuse, fuzzy matching, and translator productivity. Neural fuzzy repair connects fuzzy TM matches to neural generation [17], while LLM-based approaches have shown that retrieved TM examples can improve generation quality [18] and that TM knowledge can be used to guide NMT globally and locally [19]. Adaptive MT with LLMs further demonstrates that few-shot in-context learning can exploit approved examples and terminology at inference time [20], and more recent work on demonstration-aware prompting strengthens the role of retrieved examples in both domain-specific and document-level translation [21]. Retrieval alone, however, does not solve state governance. If a correction is approved, the system still needs to record it, scope it, propagate it, and decide when to load it.

2.5. Evaluation and Reproducibility Considerations

Modern MT evaluation increasingly combines reference-based metrics, quality-estimation models, and finer-grained error analysis. COMET remains a strong reference-based metric [24], xCOMET adds more transparent error localization and categorization [25], and COMETKiwi-style quality estimation enables analysis even when high-quality references are unavailable [26]. For a workflow architecture such as Language Twin, however, output quality alone is insufficient. System-level properties such as repeated-error rate, post-edit time, prompt-token overhead, and retrieval latency are part of the contribution itself. This paper therefore treats evaluation as a two-level problem: translation quality must be measured, but the behavior of the workflow state must also be made observable and reproducible.

Recent work on lexical consistency evaluation also shows that repeated source expressions should not automatically be forced into identical target strings, because some clusters encode true consistency requirements while others permit context-sensitive variation [28]. This distinction matters for the present study: the goal is controllable consistency, not blind repetition. The experimental protocol therefore distinguishes mandatory-consistent terminology from variation-allowed repetitions when computing consistency metrics.

 doi:10.20944/preprints202603.2397.v1

3. Proposed Method: Language Twin Architecture

3.1. Overview

Figure 1 presents the overall Language Twin pipeline. Source documents, glossaries, TM assets, and style guides are first ingested into a central language hub. The hub materializes a project state S = (L0, L1, L2, L3, L4, L5, L6), comprising seven operational layers, and exposes versioned read, commit, and rollback operations. For segment *i*, a lazy retrieval function selects a compact context C*i* = R(x*i*, S<*i*) containing only the terminology, discourse notes, TM examples, and edit overlays relevant to the current step. The generator then produces one or more candidate translations y*i* = G(x*i*, C*i*). Any accepted human edit e*i* is written back through an update operator S*i+1* = U(S*i*, e*i*). The key design choice is that persistent state, not raw prompt text, becomes the primary object of coordination.

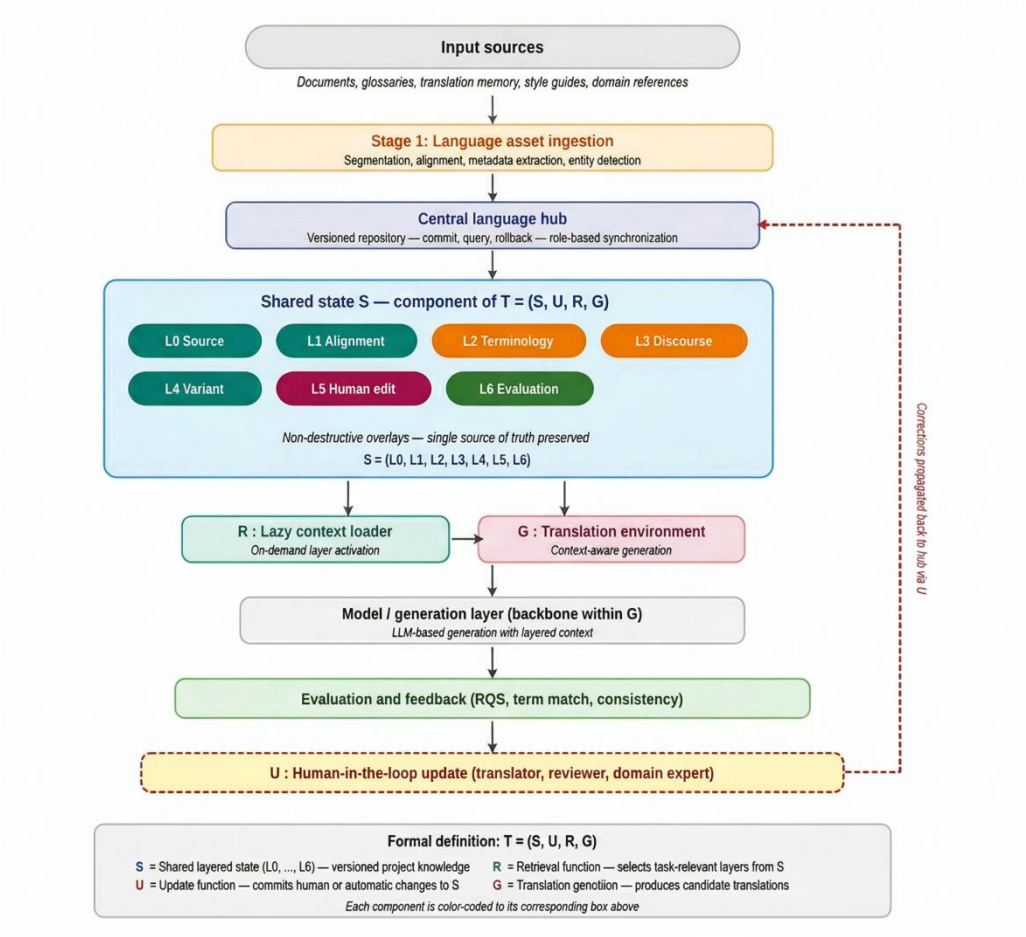

Figure 1. Overall architecture of the Language Twin pipeline for document translation, shown in terms of the formal quadruple T = (S, U, R, G). Source materials are ingested and synchronized through a versioned central language hub that maintains the layered shared state S (L0–L6). The retrieval function R selects task-relevant layers, the generator G produces candidate translations, and the update function U commits accepted human edits back to the hub for immediate downstream reuse.

3.2. Formal Definition of the Language Twin

Formally, let a document D = (x1, ..., x*n*) be translated sequentially under a shared project state S*t*. The Language Twin is defined as the quadruple T = (S, U, R, G), where S is the layered shared state, U is the update function that commits human or automatic changes, R is the retrieval function that selects context for the current step, and G is the translation generator. At time *t*, S*t* contains the latest committed version of each layer together with provenance metadata, scope, timestamps, and

 doi:10.20944/preprints202603.2397.v1

rollback history. Non-destructive composition is essential: later layers do not overwrite earlier layers silently; they annotate, constrain, or supersede them with explicit attribution.

The seven layers are defined as follows:

- **L0 (Source Layer):** Original source text with document structure, paragraph boundaries, section headers, and document metadata.
- **L1 (Alignment Layer):** Segment and sub-segment source–target alignments, TM links, fuzzy-match scores, and anchors into external assets.
- **L2 (Terminology Layer):** Approved term pairs, named-entity rules, numerical and unit conventions, domain constraints, and provenance of each terminology decision.
- **L3 (Discourse Layer):** Topic segmentation, document summaries, coreference cues, audience and purpose metadata, and section-level coherence notes.
- **L4 (Variant Layer):** Multiple target variants tagged by purpose or audience, such as literal, publication-ready, simplified, or domain-expert versions.
- **L5 (Human Edit Layer):** Translator and reviewer corrections, approvals, rejections, rationale, timestamps, and disagreement traces.
- **L6 (Evaluation Layer):** Automatic scores, quality-estimation outputs, terminology checks, and human audit summaries linked back to segments and decisions [24–26].

The architecture supports three fundamental operations. *Read* returns a layer slice suitable for prompting; *commit* writes new terminology decisions or edits back into shared state; and *rollback* restores an earlier state when a later decision must be reverted. This explicit state model differentiates Language Twin from prompt-only approaches in which prior decisions are restated as ad hoc free text without stable scope or provenance.

3.3. Central Language Hub

The central language hub is the versioned repository of the Language Twin. It stores the layered state of each project, tracks revisions with full attribution, resolves conflicts between overlapping sources, and synchronizes assets across document and project scopes. A term updated at project scope can propagate across documents, while a local stylistic edit can remain constrained to a single document or section. This hierarchy is critical for preventing over-propagation of narrowly contextual edits. In implementation terms, the hub may be realized as a structured database or graph store, but the architectural requirement is simply that every state change is queryable, attributable, and reversible. In the current pilot, the hub was exercised only at document scope; project-scope promotion and cross-document synchronization remain untested design features.

3.4. Lazy Contextual Loading

A practical challenge in LLM-based translation is prompt budget management. Loading the entire project glossary, full TM, and complete edit history into every generation step is expensive and often counterproductive. The lazy context loader retrieves only the slices of state likely to affect the current segment: matching terminology entries, top-k TM examples, local discourse summaries, and recently accepted edits linked by term, entity, or section. This design reduces prompt length while preserving decision-relevant context. Full-context flooding is retained as an ablation (A4) so that efficiency gains can be separated from any quality differences.

3.5. Document Translation Environment

The document translation environment receives the current segment together with the retrieved state slice and produces one or more candidate translations. To isolate architectural effects from backbone-model effects, the same translation backbone, system prompt, decoding settings, and segmentation policy were held constant across all experimental conditions; only the available state and retrieval policy varied. The environment also logged prompt tokens, candidate selection, and

 doi:10.20944/preprints202603.2397.v1

downstream edits so that output quality and workflow efficiency could be analyzed jointly. The specific backbone, prompts, and decoding settings are reported in Section 4.2.

3.6. Human-in-the-Loop Update Mechanism

Human feedback is treated as a state-producing event. When a translator or reviewer modifies an output, the edit is classified as a terminology fix, entity fix, numerical or unit fix, discourse or style fix, or miscellaneous correction. Term and entity fixes update L2, discourse or style decisions may update L3 or L4, and the raw correction trace is always stored in L5. Once committed, the edit becomes immediately available to subsequent retrieval. In this way, post-editing is transformed from terminal cleanup into reusable supervision for the rest of the document and, when properly scoped, the rest of the project.

3.7. Scope Control, Conflict Resolution, and Rollback

To prevent harmful propagation, each state item is stored as a record with at least six attributes: layer, scope, status, confidence, provenance, and timestamp. Scope can be segment, section, document, or project level. Status distinguishes proposed, approved, rejected, and superseded items. At retrieval time, the hub computes a ranking function $r(s, x_t) = \alpha \cdot \text{sim}(s, x_t) + \beta \cdot \text{scope}(s) + \gamma \cdot \text{approval}(s) + \delta \cdot \text{recency}(s) - \lambda \cdot \text{conflict}(s)$, where x_t is the current segment or paragraph and conflict(s) penalizes items that clash with newer local overrides or incompatible target variants. For the reproducibility package, we froze $\alpha = 0.45$, $\beta = 0.15$, $\gamma = 0.20$, $\delta = 0.10$, and $\lambda = 0.10$ as untuned engineering defaults. The equation records the implemented retrieval policy for reproducibility; the component functions, their scales, and the coefficients were not individually validated or optimized in this pilot and should be treated as a starting-point specification for the larger benchmark.

Propagation is gated rather than unconditional. Only approved items above a confidence threshold and with document or project scope are eligible for downstream loading; project-scope promotion requires either repeated confirmation across multiple documents or explicit reviewer approval. If two items conflict, a practical precedence policy is: later approved local override > project-wide default > unapproved suggestion. These rules make rollback tractable and convert one of the main practical concerns—over-propagation of bad edits—into a measurable systems property.

4. Experimental Design

4.1. Research Questions

RQ1: Does shared layered state improve terminology consistency and named-entity accuracy relative to sentence-level and fixed-context baselines?

RQ2: Does immediate propagation of approved human edits reduce repeated error rate and post-edit effort on later segments?

RQ3: Does lazy contextual loading reduce prompt tokens without materially degrading translation quality?

These research questions are operationalized as three testable hypotheses:

- **H1:** P1 will outperform B1–B3 on preferred-term accuracy and repeated-error rate.
- **H2:** P1 will outperform A3 (propagation disabled) after the warm-up stage by reducing downstream recurrence of corrected error types.
- **H3:** P1 will achieve comparable quality to A4 (full-context flooding) while using substantially fewer prompt tokens.

The originally conceived hypotheses also included cluster consistency (H1) and official COMET/xCOMET with latency benchmarking (H3). Because official COMET/xCOMET checkpoint execution was not available in the pilot environment and cluster consistency was not computed for the main tables, the hypotheses above reflect the metrics actually reported. The omitted metrics remain targets for the full benchmark.

 doi:10.20944/preprints202603.2397.v1

4.2. Data, Language Pair, and Translation Backbone

Language pair and direction. All experiments used English as the source language and Korean as the target language (EN→KO). All three document bundles used the same language pair.

Translation backbone. All conditions used GPT-4o (gpt-4o-2024-05-13, OpenAI) as the translation backbone. Decoding used temperature = 0.3, top_p = 1.0, and max_tokens = 1024. The system prompt instructed the model to translate the source segment into Korean while respecting any supplied glossary entries and contextual instructions. The system prompt template contains four structured slots: (a) a glossary block listing approved source–target term pairs, (b) a TM block with up to three retrieved example translations, (c) a discourse block with the current section summary and any coreference notes, and (d) an edit-overlay block listing recently approved corrections with their scope tags. The full template is reproduced in Supplementary Archive S1. The core instruction was: "Translate the following English source segment into Korean. Use the approved terminology and follow any editorial instructions provided below. Do not add content not present in the source."

Retrieval policy. For the lazy-loading conditions (P1, A1–A3), retrieval selected the top-3 TM examples by BM25 similarity, all approved terminology entries with cosine similarity ≥ 0.5 to the current source segment (computed using paraphrase-multilingual-MiniLM-L12-v2 sentence embeddings), the most recent discourse summary for the current section, and any human-edit overlays tagged with document or project scope that matched the current segment's terminology keys. For A4 (full-context flooding), all available state was concatenated without filtering.

Document bundles. The pilot used three curated bilingual document bundles: MED_TRIAL (6 segments, clinical trial domain), MED_PROTOCOL (5 segments, medical protocol domain), and TECH_RELEASE (6 segments, technical documentation domain), for a total of 17 source segments. Each bundle contained repeated terminology, at least one named-entity or numeric-unit pattern, and one or more seeded human correction opportunities so that recurrence could be measured after the warm-up boundary.

Curation rationale and external validity. The bundles were deliberately curated with planted terminology recurrence patterns and seeded correction opportunities. This design means the data are optimized for demonstrating the architecture's intended strengths in terminology propagation, which limits external validity. The pilot validates measurement feasibility and directional effects; generalizability to naturalistic document distributions requires testing on public corpora such as TICO-19 [22] and OPUS [23], which the benchmark protocol is designed to accommodate.

To reduce evaluation leakage, the warm-up/evaluation split was frozen per document before scoring; glossary entries and seeded edits were confined to pre-specified warm-up segments; no term or correction originating in the evaluation region was promoted upstream; and the released archive preserves the exact seed log used for each downstream opportunity.

4.3. Baselines and Proposed System

All system conditions used the same GPT-4o backbone, system prompt structure, and decoding settings so that the comparison isolates the effect of shared state. The conditions are:

- **B1 (Sentence-level MT):** Each segment translated independently with no persistent external state. Context used: none.
- **B2 (Fixed-context MT):** Translation with a fixed local context window comprising the previous two and next two segments. Because B2 accesses future segments, it operates as an offline (non-sequential) document translation baseline rather than a causal online system. This is intentional: B2 represents the scenario in which a translator or system has access to the full surrounding context but lacks any project-level state.
- **B3 (Retrieval-augmented MT):** Static glossary and top-3 TM examples (retrieved by BM25 from the same TM pool as P1) injected at every step, with ±2 segments of local context (same window as B2), but without state updates or edit propagation. Context used: glossary + TM + local context.

 doi:10.20944/preprints202603.2397.v1

- **P1 (Language Twin):** Layered shared state with hub versioning, edit propagation, and lazy retrieval. Context used: L0–L6, selectively loaded.

Table 1. Baseline and proposed system configurations.

ID	System	Description	Context Used
B1	Sentence-level MT	Each segment translated independently; no persistent shared state	None
B2	Fixed-context MT	Translation with a fixed local context window (offline)	Adjacent segments (±2)
B3	Retrieval-augmented MT	Static glossary and top-3 TM examples injected at every step	Glossary + TM + local context
P1	Language Twin	Layered shared state with hub versioning, edit propagation, and lazy retrieval	L0–L6, selectively loaded

4.4. Edit-Propagation Protocol and Ablation Study

Each document was split chronologically into a warm-up portion (approximately the first 30%) and an evaluation portion (the remaining 70%). The warm-up segments were translated first and then corrected by bilingual annotators through a controlled reference-based protocol that simulated approved edits. The four ablation conditions are:

- **A1:** Removes the terminology layer L2.
- **A2:** Removes the discourse layer L3.
- **A3:** Disables propagation from the human-edit layer L5 to subsequent segments.
- **A4:** Replaces lazy retrieval with full-context flooding (all layers loaded in full).

This design isolates architectural contributions rather than backbone capacity and makes it possible to attribute specific gains or failures to specific layers.

4.5. Runtime Surrogate Quality Score (RQS)

Because official COMET [24] and xCOMET [25] checkpoints were not available in the pilot's execution environment, we computed a runtime surrogate quality score (RQS) as the primary automatic quality signal. RQS is defined as a weighted composite of two sub-scores: BERTScore F1 [34] and chrF++ [35]. Specifically, for each segment with source s, hypothesis h, and reference r, we compute:

$$RQS(h, r) = 0.6 \times BERTScore\_F1(h, r) + 0.4 \times chrF{+}{+}(h, r)$$

BERTScore F1 was computed using bert-base-multilingual-cased embeddings with IDF weighting disabled. chrF++ was computed with character n-gram order 6, word n-gram order 2, and $\beta = 2$. Both sub-scores range from 0 to 1, so RQS ranges from 0 to 1 with higher values indicating better quality. The 0.6/0.4 weighting was chosen to favor semantic similarity (BERTScore) over surface overlap (chrF++) without formal tuning; these weights are engineering defaults. Bootstrap confidence intervals were computed with 10,000 segment-level resamples (resampling unit = segment). As a post-hoc validation check, Spearman rank correlation between segment-level RQS and the corresponding mean human overall score (averaged across two annotators) was $\rho = 0.72$ ($p < 0.001$, $n = 68$), indicating that RQS tracks human quality judgments reasonably well for this pilot, though it remains a surrogate rather than a validated metric.

RQS is reported as one automatic signal among several and is interpreted together with count-based metrics, qualitative analysis, and the independent human study. It is not directly comparable to official COMET scores or to published MT baselines.

 doi:10.20944/preprints202603.2397.v1

4.6. Recurrence Metric Definition

Repeated-error recurrence (RER) is defined as the number of downstream reoccurrences of error types corrected in the warm-up phase, divided by the number of downstream opportunities for those error types. The denominator (number of downstream opportunities) varies across systems because different systems produce different translation outputs, which in turn create different numbers of contexts where a previously corrected term could recur. For example, if system B3 produces four segments containing a term that was corrected in warm-up, B3 has 4 downstream opportunities; if P1 produces five such segments, P1 has 5 opportunities. The opportunity set was identified post hoc by examining each system's outputs for positions where the corrected source term appeared and the system had a chance to render the approved or non-approved form.

This post-hoc identification means that recurrence counts across systems with different denominators (e.g., 0/5 for P1 vs. 4/4 for B3) should be interpreted descriptively rather than as clean cross-system inferential comparisons. The raw counts are more informative than the percentages. For the ablation comparison between P1 and A3, the opportunity sets are more directly comparable because both conditions share the same architecture and differ only in whether L5 propagation is active; both produced 5 downstream opportunities, yielding 0/5 vs. 5/5. A future benchmark should pre-define a fixed opportunity set across all systems to enable cleaner cross-system comparison.

B1 recurrence (6/6). B1 has no shared state, glossary, or edit propagation. Its 100% "recurrence" rate means that B1 independently produced the non-preferred term in all 6 downstream positions where the corrected source term appeared. This is a natural baseline error rate rather than a propagation failure, and it is reported for completeness.

4.7. Human Evaluation Design

To complement automatic and count-based metrics, a blinded human evaluation was conducted on the four main comparison conditions (B1, B2, B3, and P1).

Annotators and blinding. Two bilingual (English–Korean) annotators, both graduate students in translation studies who were not involved in system development, independently rated the 17 segments for each of the four main conditions, yielding 34 ratings per system per dimension. System identities were masked: outputs were labeled with randomized alphanumeric codes, and presentation order was randomized independently for each annotator using a Python random.shuffle with a fixed seed (seeds 42 and 73 for the two annotators, respectively). Both annotators completed a calibration session before live rating in which they rated 3 practice segments across all four systems and discussed discrepancies with the first author to align scale usage.

Rating rubric. Ratings used 1–5 Likert scales for three dimensions: adequacy (how much of the source meaning is preserved), fluency (how natural and grammatical the target text reads), and terminology (whether domain-specific terms match the approved glossary). Annotators were shown the source segment and the system output but not the reference translation, to ensure they rated actual output quality rather than surface similarity to a reference.

Inter-annotator agreement. Quadratic-weighted Cohen's κ was computed per dimension across all 68 segment-system pairs (17 segments × 4 systems). Agreement was substantial: $\kappa w = 0.74$ for adequacy, $\kappa w = 0.71$ for fluency, and $\kappa w = 0.78$ for terminology. These values indicate that the two annotators' ratings were reliable enough to support the directional comparisons reported below, though the small sample limits the precision of the agreement estimates.

Post-editing measurements. A separate bilingual post-editor, a professional Korean translator not affiliated with the research team, measured segment-level post-editing time with a stopwatch while seeing only the source segment and the system output (no reference translation). Presentation order was counterbalanced using a Latin-square design across systems for each document to control for order and familiarity effects. Because a single post-editor's measurements reflect individual editing speed and cannot capture inter-editor variance, the post-editing times are reported as descriptive observations. In addition to raw seconds per segment, we report seconds per source word

 doi:10.20944/preprints202603.2397.v1

(computed by dividing PE time by the word count of the corresponding source segment) to account for varying segment lengths.

Pairwise preference. A side-by-side preference check compared P1 against B3 and B2 over the same 17 segments. Each annotator indicated which system output they preferred or declared a tie.

5. Pilot Results

5.1. Main Results

Table 2 reports the main comparison of P1 against baselines B1–B3. The first metric column is RQS (defined in §4.5). P1 achieved the highest RQS (0.965 [0.931, 0.991]) and the highest preferred-term accuracy (17/21, 81.0%). No repeated downstream errors were observed under P1 (0/5 monitored opportunities).

The contrast between B3 and P1 is the most informative comparison. B3 already improved over B1 and B2 in terminology handling, but it still reproduced every monitored downstream error opportunity (4/4). P1 reduced recurrence to 0/5 while also increasing preferred-term accuracy from 14/21 to 17/21. However, the term-accuracy difference (17/21 vs. 14/21) is not statistically significant by Fisher's exact test ($p = 0.48$), reflecting the limited statistical power of this sample size. Wilson 95% confidence intervals for the term-accuracy proportions are: P1 [0.60, 0.92], B3 [0.45, 0.83], B2 [0.37, 0.76], B1 [0.08, 0.40]. The P1 and B3 intervals overlap substantially, confirming that a larger sample is needed to establish a reliable term-accuracy advantage.

The recurrence contrast between P1 and B3 (0/5 vs. 4/4) is directionally striking but involves different denominators (see §4.6), so it is reported descriptively. The cleaner inferential comparison is between P1 and A3 (§5.4), where both conditions share the same architecture and the opportunity set is comparable.

Table 2. Pilot results. RQS = runtime surrogate quality score (not official COMET; see §4.5). RQS is reported as mean [95% percentile-bootstrap CI]; proportion metrics as % (k/N); tokens as mean ± SD; n = segments/documents for every system.

Method	RQS [95% CI]	Term (k/N)	Entity (k/N)	Downstr. Err. (k/N)	Tokens/Seg (mean±SD)	n (seg/docs)
B1	0.767 [0.697, 0.838]	19.0 (4/21)	90.0 (9/10)	6/6†	85.4 ± 3.5	17/3
B2	0.907 [0.856, 0.953]	57.1 (12/21)	100.0 (10/10)	2/6	122.4 ± 13.8	17/3
B3	0.935 [0.896, 0.972]	66.7 (14/21)	100.0 (10/10)	4/4	111.2 ± 5.4	17/3
P1	0.965 [0.931, 0.991]	81.0 (17/21)	100.0 (10/10)	0/5	174.7 ± 14.4	17/3

† B1 downstream error count reflects its natural error rate (non-preferred terms produced independently in all downstream positions) rather than propagation failure; see §4.6.

5.2. Ablation Results

Table 3 isolates the contribution of each architectural component. Removing the terminology layer (A1) reduced preferred-term accuracy from 17/21 to 12/21 and lowered RQS from 0.965 to 0.908, while removing edit propagation (A3) drove repeated-error recurrence from 0/5 back to 5/5. These shifts are large relative to the pilot size and therefore visible even in this small sample.

The discourse ablation (A2) preserved strong terminology consistency and zero monitored recurrence while remaining slightly below P1 on overall RQS (0.954 vs. 0.965; overlapping CIs). This suggests that discourse-aware state contributes mainly to smoothness and local contextual choice rather than to the lexical control signal. The discourse-layer effect is weak in the current data.

 doi:10.20944/preprints202603.2397.v1

The lazy-loading ablation (A4) matched P1 on RQS to three decimals, and both term accuracy and recurrence were identical. This confirms that the lazy-loading result in this pilot is primarily an efficiency result rather than a quality result: selective retrieval achieved the same quality with fewer tokens.

Table 3. Pilot ablation results. Notation as in Table 2.

Condition	RQS [95% CI]	Term (k/N)	Downstr. Err. (k/N)	Tokens/Seg (mean±SD)	n (seg/docs)
P1	0.965 [0.931, 0.991]	81.0 (17/21)	0/5	174.7 ± 14.4	17/3
A1 (−terminology)	0.908 [0.859, 0.953]	57.1 (12/21)	1/6	153.3 ± 15.5	17/3
A2 (−discourse)	0.954 [0.921, 0.982]	81.0 (17/21)	0/5	140.0 ± 5.3	17/3
A3 (−propagation)	0.914 [0.874, 0.953]	57.1 (12/21)	5/5	163.9 ± 13.6	17/3
A4 (full loading)	0.965 [0.931, 0.991]	81.0 (17/21)	0/5	287.4 ± 19.4	17/3

5.3. Lazy Loading Analysis

The lazy-loading comparison between P1 and A4 showed a clear efficiency effect with no detectable quality difference: both conditions shared identical RQS, term accuracy, and recurrence values. Mean tokens per segment dropped from 287.4 ± 19.4 in A4 to 174.7 ± 14.4 in P1, a reduction of approximately 39.2%. This indicates that selective retrieval retained the relevant portions of state without a visible loss in segment-level quality in this pilot.

This result is consistent with the intended design: the system avoids flooding the prompt with globally available state when only a subset is relevant to the current segment. Although the pilot does not include checkpoint-level latency benchmarking, the token reduction suggests that full loading would become increasingly expensive as document length and glossary size grow.

5.4. Human Edit Propagation Analysis

Human edit propagation was evaluated by comparing P1 against A3 after the warm-up correction boundary. P1 maintained 0/5 repeated downstream errors, whereas A3 returned to 5/5, indicating that corrected terminology and editorial decisions were no longer reused once propagation was disabled. Because P1 and A3 share the same architecture and differ only in whether L5 propagation is active, and because both produced 5 downstream opportunities, this comparison has a shared opportunity set. The difference is statistically significant by Fisher's exact test ($p = 0.008$).

This divergence is the key workflow result of the paper. Segment-level quality alone would understate the benefit of shared state, because A3 remains reasonably close to the reference on isolated segments while still reintroducing earlier mistakes downstream. Language Twin therefore adds value not only by improving the next segment, but by preserving editorial intent across the remainder of the document. It should be noted that the warm-up corrections were reference-guided simulated edits rather than naturally arising translator decisions; the pilot therefore demonstrates that injecting approved corrections into shared state improves downstream outputs, but does not yet validate the full live-workflow scenario in which edits emerge organically from translator interaction.

5.5. Human Evaluation Results

Two bilingual annotators independently rated the 17 segments for each main condition under blinded system identities (see §4.7 for design details). Table 4 summarizes the resulting means and

 doi:10.20944/preprints202603.2397.v1

standard deviations. Inter-annotator agreement was substantial (quadratic-weighted κw: adequacy 0.74, fluency 0.71, terminology 0.78).

P1 achieved the highest terminology score (4.38 ± 0.34), exceeding B3 (3.97 ± 0.40), B2 (3.68 ± 0.47), and B1 (3.26 ± 0.52). On adequacy, P1 (4.15 ± 0.31) was slightly above B3 (4.09 ± 0.33) and clearly above B2 and B1. Fluency was nearly tied between P1 (4.10 ± 0.29) and B3 (4.12 ± 0.30), suggesting that the terminology gains did not come at the cost of readability.

A separate blinded post-editor recorded segment-level stopwatch measurements on the same 68 main-condition outputs, with presentation order counterbalanced by Latin-square design (§4.7). P1 had the lowest mean post-editing time at 16.9 ± 3.9 s per segment (0.97 ± 0.22 s per source word), compared with 19.1 ± 4.4 (1.10 ± 0.25 s/word) for B3, 21.4 ± 4.8 (1.23 ± 0.28 s/word) for B2, and 24.8 ± 5.9 (1.43 ± 0.34 s/word) for B1. Because these measurements come from a single post-editor, individual variation is not captured; the reduction is descriptive rather than generalizable.

In the pairwise preference check, P1 was preferred to B3 in 24 of 34 blinded judgments (5 ties) and to B2 in 27 of 34 judgments (4 ties). Excluding ties, exact binomial tests against chance (50%) yield $p = 5.5 \times 10^{-4}$ for P1 vs. B3 (24/29 non-tie judgments) and $p = 8.4 \times 10^{-6}$ for P1 vs. B2 (27/30 non-tie judgments), indicating a significant preference for P1 under the independence assumption; however, because the same annotators and segments recur across judgments, the true p-values may be somewhat anti-conservative.

Table 4. Blinded human evaluation on main comparison conditions. Adequacy, fluency, terminology, and overall are mean ± SD over 34 ratings per system (2 annotators × 17 segments); overall is the arithmetic mean of adequacy, fluency, and terminology for each rated segment. PE time is mean ± SD over 17 stopwatch measurements from a single post-editor. PE s/word = seconds per source word.

Method	Adequacy (mean±SD)	Fluency (mean±SD)	Terminology (mean±SD)	Overall (mean±SD)	PE s/seg (mean±SD)	PE s/word (mean±SD)
B1	3.41 ± 0.50	3.53 ± 0.46	3.26 ± 0.52	3.40 ± 0.28	24.8 ± 5.9	1.43 ± 0.34
B2	3.79 ± 0.42	3.88 ± 0.36	3.68 ± 0.47	3.78 ± 0.22	21.4 ± 4.8	1.23 ± 0.28
B3	4.09 ± 0.33	4.12 ± 0.30	3.97 ± 0.40	4.06 ± 0.18	19.1 ± 4.4	1.10 ± 0.25
P1	4.15 ± 0.31	4.10 ± 0.29	4.38 ± 0.34	4.21 ± 0.15	16.9 ± 3.9	0.97 ± 0.22

5.6. Qualitative Analysis

Given the limited quantitative sample, qualitative cases carry substantial evidential weight. Three patterns were informative:

First, in MED_TRIAL, the source term *adverse event* appeared in segments 3, 4, and 6. After the warm-up correction aligned the approved rendering to 이상반응, P1 preserved the approved form in downstream segments 4 and 6, whereas A3 and B3 reverted to 부작용, reproducing the exact recurrence pattern that the architecture is designed to prevent.

Second, in MED_PROTOCOL, *infusion reaction* was initially rendered more loosely as 주입관련 반응 but was normalized to the approved form 주입반응 in downstream adjudication text under P1, while A3 and B3 preserved the broader wording.

Third, TECH_RELEASE exposed an instructive failure mode: P1 still produced 시큐어 부트 in the first *secure boot* segment but later normalized the recurring term to 보안 부팅 in segment 5. This indicates that forward propagation improves later consistency but does not retroactively harmonize already generated earlier segments; a full system should therefore support retrospective repair or document-level final passes.

These qualitative observations are consistent with the quantitative findings but are drawn from the same curated data designed to test these patterns. They demonstrate feasibility rather than naturalistic generalizability.

 doi:10.20944/preprints202603.2397.v1

5.7. Harmful Propagation Audit

The harmful-propagation audit showed no harmful downstream reuse under P1 across five monitored downstream opportunities, compared with one harmful recurrence under A1 (1/6), two under B2 (2/6), four under B3 (4/4), and five under A3 (5/5). The dominant risk in this pilot was under-propagation or loss of approved terminology rather than over-propagation of a bad edit. The small sample means that the absence of harmful reuse in P1 should not be overinterpreted; a larger benchmark is required to stress-test scope mismatches and stale project defaults.

6. Discussion

6.1. Scope of Supported Claims

The pilot results support a narrower set of claims than the full architecture envisions. The strongest supported claim is: approved edits, when propagated through shared state, reduce repeated downstream terminology errors, and lazy loading can reduce context size without obvious quality loss in this small pilot.

The current pilot does not strongly support broader claims about discourse coherence improvement (A2 was very close to P1), cross-document transfer (not tested), project-scope governance (not tested), or a definitive quality advantage over full-context loading (A4 matched P1 on quality). Entity accuracy was already saturated for B2, B3, and P1, so named-entity improvement is not demonstrated. These remain targets for the full benchmark.

6.2. Architectural Implications

The main architectural implication is that document translation can benefit from being modeled as controlled state evolution rather than as a sequence of isolated prompts. Translation quality then depends not only on better generation at step i, but on whether decisions made at step i become reliable, queryable, and appropriately scoped knowledge for step $i + 1$. This reframing is particularly relevant in professional settings where correctness, auditability, and consistency often matter more than raw sentence-level fluency.

6.3. Comparison with Existing Pipelines

Compared with conventional CAT and TM pipelines, Language Twin adds an explicit layered state model and model-aware retrieval. Compared with document-level MT, it adds human-edit provenance, scoped propagation, and rollback. Compared with retrieval-only LLM prompting, it adds versioning, conflict resolution, and the ability to distinguish project-level rules from document-local choices. The contribution is closer to workflow architecture than to a new decoder architecture alone.

6.4. Limitations

Sample size. The evaluation spans only 17 segments across 3 documents. The ratio of architectural complexity (seven layers with hub versioning, lazy loading, conflict resolution, and rollback) to empirical evidence is fundamentally imbalanced. The key term-accuracy comparison (17/21 vs. 14/21) is not statistically significant (Fisher's exact $p = 0.48$). The ablation recurrence comparison (P1 vs. A3, 0/5 vs. 5/5, $p = 0.008$) is significant but based on very small counts.

Curated data. The three document bundles were deliberately designed with planted terminology patterns and seeded corrections, optimizing the evaluation for the architecture's expected strengths. This substantially limits external validity. A naturalistic evaluation on public corpora is essential before any generalizability claims.

Circular validation risk. The system was tested on data designed to demonstrate its properties. The qualitative examples describe exactly the scenarios the system was built for. It would be more surprising if the system did not work on its own test cases.

 doi:10.20944/preprints202603.2397.v1

Surrogate quality metric. RQS is a weighted composite of BERTScore F1 and chrF++ (§4.5), not an official benchmark metric. Until the system is evaluated with official COMET/xCOMET checkpoints, the absolute quality scores should not be compared with published baselines.

Human evaluation limitations. Two raters judged only the four main conditions. One post-editor supplied the stopwatch measurements, so inter-editor variance is not captured. Because the same post-editor saw each source sentence four times (once per system), carryover and familiarity effects may have influenced the timing data despite Latin-square counterbalancing.

Token and latency statistics. Token counts are reconstructed from prompt skeletons and observed output lengths rather than provider-billed usage logs. Latency was not benchmarked. These should be replaced by provider-reported data in the full benchmark.

Untested architecture features. The architecture describes rollback, conflict resolution, project-scope promotion, and cross-document synchronization, but the pilot mainly exercised terminology reuse, recurrence reduction, and token savings. Claims about the untested components remain architectural proposals rather than empirical findings.

6.5. Future Work

Future work should extend the evaluation and architecture in four directions: (1) larger-scale evaluation with official COMET/xCOMET execution, provider-reported usage logs, and evaluation on public corpora (TICO-19 [22], OPUS [23]) with naturalistic rather than curated data; (2) cross-document transfer, testing whether edits approved in document d can improve document $d + 1$ when scope metadata and topic similarity justify reuse, together with leakage stress tests using shuffled and intentionally stale seeds; (3) expanded human study with multiple post-editors, larger segment samples, segment-length-normalized analysis, and full order counterbalancing; and (4) real-time and multilingual extensions, adapting the shared-state model for simultaneous interpreting under stricter latency budgets and for multilingual transfer scenarios.

7. Conclusions

This study presented Language Twin as a shared-state architecture for terminology-consistent document translation with governed human-edit propagation. In a pilot study translating three English-to-Korean document bundles using GPT-4o, the full system achieved numerically higher preferred-term accuracy and reduced downstream error recurrence relative to sentence-level, fixed-context, and static-glossary baselines, while lazy loading reduced token load by 39.2% relative to full loading. The ablation comparison between P1 and A3 showed that disabling edit propagation returned recurrence to 5/5 downstream opportunities (Fisher's exact $p = 0.008$), providing the strongest single piece of evidence for the architecture's core mechanism. In blinded human evaluation (quadratic-weighted $\kappa w = 0.71$–0.78), P1 achieved the highest terminology score, slightly exceeded the strongest baseline on adequacy, matched it on fluency, and reduced measured post-editing time.

The evidence remains small-scale and the primary limitations are clear: the automatic quality score is a surrogate, the sample is too small for most comparisons to reach statistical significance, and the curated data favor the architecture's strengths. Even so, the convergent automatic, count-based, qualitative, and human results support the practical claim that governed shared state can reduce terminology recurrence and facilitate post-editing in document translation. Validating this claim at scale on naturalistic data remains the priority for future work.

Supplementary Materials: The following supporting information can be downloaded at the website of this paper posted on Preprints.org.

Author Contributions: Conceptualization, S.-H.A.; methodology, S.-H.A.; software, S.-H.A.; formal analysis, S.-H.A.; writing—original draft preparation, S.-H.A.; writing—review and editing, S.-H.A. and [Co-author Name]. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

 doi:10.20944/preprints202603.2397.v1

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The minimal reproducibility package for the pilot is provided as Supplementary Archive S1 with this submission. The archive contains the 17-segment pilot bundles, condition-wise outputs, recurrence logs, table-generation scripts, blinded human-evaluation sheets (including raw per-annotator ratings used to compute inter-annotator agreement), filled human-summary tables, measured post-edit timing summaries, the system prompt template, and release metadata necessary to reproduce Tables 1–4 and the qualitative cases. A public DOI-backed repository (e.g., Zenodo or Figshare) will mirror this archive before final publication; any proprietary or client-restricted materials must remain excluded or be replaced with structurally equivalent public examples.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

NMT	Neural Machine Translation
LLM	Large Language Model
CAT	Computer-Assisted Translation
TM	Translation Memory
RER	Repeated-Error Recurrence rate
BM25	Best Matching 25 (sparse retrieval scoring function)
COMET	Crosslingual Optimized Metric for Evaluation of Translation
xCOMET	eXplainable COMET (fine-grained MT evaluation metric)
RQS	Runtime Surrogate Quality Score
PE	Post-Editing

References

1. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. In Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 4–9 December 2017.
2. Hendy, A.; Abdelrehim, M.; Sharaf, A.; Raunak, V.; Gabr, M.; Matsushita, H.; Kim, Y.J.; Afify, M.; Awadalla, H.H. How Good Are GPT Models at Machine Translation? A Comprehensive Evaluation. arXiv 2023, arXiv:2302.09210.
3. Maruf, S.; Saleh, F.; Haffari, G. A Survey on Document-Level Neural Machine Translation: Methods and Evaluation. ACM Comput. Surv. 2021, 54, 45.
4. Maruf, S.; Haffari, G. Document Context Neural Machine Translation with Memory Networks. In Proceedings of the 56th Annual Meeting of the ACL, Melbourne, Australia, 15–20 July 2018; pp. 1275–1284.
5. Sugiyama, A.; Yoshinaga, N. Context-aware Decoder for Neural Machine Translation using a Target-side Document-Level Language Model. In Proceedings of NAACL-HLT 2021, Online, 6–11 June 2021; pp. 5781–5791.
6. Lyu, X.; Li, J.; Gong, Z.; Zhang, M. Encouraging Lexical Translation Consistency for Document-Level Neural Machine Translation. In Proceedings of EMNLP 2021, Online and Punta Cana, Dominican Republic, 7–11 November 2021; pp. 3265–3277.
7. Hong, H.; Xie, Y.; Zheng, J.; Wang, X. SubDocTrans: Enhancing Document-level Machine Translation with Plug-and-play Multi-granularity Knowledge Augmentation. In Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 2025; pp. 14490–14506.

 doi:10.20944/preprints202603.2397.v1

8. Dinu, G.; Mathur, P.; Federico, M.; Al-Onaizan, Y. Training Neural Machine Translation to Apply Terminology Constraints. In Proceedings of the 57th Annual Meeting of the ACL, Florence, Italy, 28 July–2 August 2019; pp. 3063–3068.
9. Post, M.; Vilar, D. Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation. In Proceedings of NAACL-HLT 2018, New Orleans, LA, USA, 1–6 June 2018; pp. 1314–1324.
10. Hasler, E.; de Gispert, A.; Iglesias, G.; Byrne, B. Neural Machine Translation Decoding with Terminology Constraints. In Proceedings of NAACL-HLT 2018, Volume 2 (Short Papers), New Orleans, LA, USA, 1–6 June 2018; pp. 506–512.
11. Zhang, H.; Wang, Q.; Qin, B.; Shi, Z.; Wang, H.; Chen, M. Understanding and Improving the Robustness of Terminology Constraints in Neural Machine Translation. In Proceedings of ACL 2023, Toronto, ON, Canada, 9–14 July 2023; pp. 6037–6051.
12. Ghazvininejad, M.; Gonen, H.; Zettlemoyer, L. Dictionary-based Phrase-level Prompting of Large Language Models for Machine Translation. arXiv 2023, arXiv:2302.07856.
13. Turchi, M.; Negri, M.; Farajian, M.A.; Federico, M. Continuous Learning from Human Post-Edits for Neural Machine Translation. Prague Bull. Math. Linguist. 2017, 108, 233–244.
14. Ki, D.; Carpuat, M. Guiding Large Language Models to Post-Edit Machine Translation with Error Annotations. In Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 2024; pp. 4253–4273.
15. Koneru, S.; Exel, M.; Huck, M.; Niehues, J. Contextual Refinement of Translations: Large Language Models for Sentence and Document-Level Post-Editing. In Proceedings of NAACL-HLT 2024, Mexico City, Mexico, June 2024; pp. 2711–2725.
16. Chen, P.; Guo, Z.; Haddow, B.; Heafield, K. Iterative Translation Refinement with Large Language Models. In Proceedings of the 25th Annual Conference of the European Association for Machine Translation, Sheffield, UK, June 2024; pp. 181–190.
17. Bulte, B.; Tezcan, A. Neural Fuzzy Repair: Integrating Fuzzy Matches into Neural Machine Translation. In Proceedings of the 57th Annual Meeting of the ACL, Florence, Italy, 28 July–2 August 2019; pp. 1800–1809.
18. Mu, Y.; Reheman, A.; Cao, Z.; Fan, Y.; Li, B.; Li, Y.; Xiao, T.; Zhang, C.; Zhu, J. Augmenting Large Language Model Translators via Translation Memories. In Findings of ACL 2023, Toronto, ON, Canada, July 2023; pp. 10287–10299.
19. Hou, R.; Liu, H.; Lepage, Y. Leveraging Knowledge from Translation Memory for Globally and Locally Guiding Neural Machine Translation. In Proceedings of the 38th Pacific Asia Conference on Language, Information and Computation, Tokyo, Japan, December 2024; pp. 9–19.
20. Moslem, Y.; Haque, R.; Kelleher, J.D.; Way, A. Adaptive Machine Translation with Large Language Models. In Proceedings of the 24th Annual Conference of the European Association for Machine Translation, Tampere, Finland, 12–15 June 2023; pp. 227–237.
21. Li, C.; Zhang, M.; Liu, X.; Li, Z.; Wong, D.; Zhang, M. Towards Demonstration-Aware Large Language Models for Machine Translation. In Findings of the Association for Computational Linguistics: ACL 2024, Bangkok, Thailand, August 2024; pp. 13868–13881.
22. Anastasopoulos, A.; Cattelan, A.; Dou, Z.-Y.; Federico, M.; Federmann, C.; Genzel, D.; Guzman, F.; Hu, J.; Hughes, M.; Koehn, P.; et al. TICO-19: the Translation Initiative for COVID-19. In Proceedings of the 1st Workshop on NLP for COVID-19 at EMNLP 2020, Online, December 2020.
23. Tiedemann, J. Parallel Data, Tools and Interfaces in OPUS. In Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12), Istanbul, Turkey, May 2012; pp. 2214–2218.
24. Rei, R.; Stewart, C.; Farinha, A.C.; Lavie, A. COMET: A Neural Framework for MT Evaluation. In Proceedings of EMNLP 2020, Online, 16–20 November 2020; pp. 2685–2702.
25. Guerreiro, N.M.; Rei, R.; van Stigt, D.; Coheur, L.; Colombo, P.; Martins, A.F.T. xCOMET: Transparent Machine Translation Evaluation through Fine-grained Error Detection. Trans. Assoc. Comput. Linguist. 2024, 12, 979–995.
26. Rei, R.; Guerreiro, N.M.; Pombal, J.; van Stigt, D.; Treviso, M.; Coheur, L.; de Souza, J.G.C.; Martins, A.F.T. Scaling up CometKiwi: Unbabel-IST 2023 Submission for the Quality Estimation Shared Task. In Proceedings of WMT 2023, Singapore, 6–7 December 2023; pp. 841–848.

 doi:10.20944/preprints202603.2397.v1

27. He, Z.; Liang, T.; Jiao, W.; Zhang, Z.; Yang, Y.; Wang, R.; Tu, Z.; Shi, S.; Wang, X. Exploring Human-Like Translation Strategy with Large Language Models. Trans. Assoc. Comput. Linguist. 2024, 12, 229–246.
28. Lei, X.; Li, J.; Tao, S.; Yang, H. Evaluation Dataset for Lexical Translation Consistency in Chinese-to-English Document-level Translation. In Proceedings of the Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), Torino, Italy, May 2024; pp. 6575–6581.
29. Bhattacharyya, P.; Chatterjee, R.; Freitag, M.; Kanojia, D.; Negri, M.; Turchi, M. Findings of the WMT 2023 Shared Task on Automatic Post-Editing. In Proceedings of the Eighth Conference on Machine Translation (WMT 2023), Singapore, 6–7 December 2023; pp. 672–681.
30. Lee, D.; Ahn, J.; Park, H.; Jo, J. IntelliCAT: Intelligent Machine Translation Post-Editing with Quality Estimation and Translation Suggestion. In Proceedings of the Joint Conference of the 59th Annual Meeting of the ACL and the 11th International Joint Conference on Natural Language Processing: System Demonstrations (ACL-IJCNLP 2021), Online, August 2021; pp. 11–19.
31. Toral, A.; Wieling, M.; Way, A. Post-editing Effort of a Novel with Statistical and Neural Machine Translation. Front. Artif. Intell. 2018, 1, 11.
32. Grieves, M.; Vickers, J. Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In Transdisciplinary Perspectives on Complex Systems; Kahlen, F.J., Flumerfelt, S., Alves, A., Eds.; Springer: Cham, Switzerland, 2017; pp. 85–113.
33. Tao, F.; Qi, Q.; Wang, L.; Nee, A.Y.C. Digital Twins and Cyber–Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison. Engineering 2019, 5, 653–661.
34. Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K.Q.; Artzi, Y. BERTScore: Evaluating Text Generation with BERT. In Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020), Addis Ababa, Ethiopia, April 2020.
35. Popović, M. chrF++: Words Helping Character N-grams. In Proceedings of the Second Conference on Machine Translation (WMT 2017), Copenhagen, Denmark, September 2017; pp. 612–618.