

Retrieval Without Relevance: Prompt Dependence in Small Machine Translation Models

Heesu Kim¹, Shota Sato², Mamoru Komachi²

¹Sungkyunkwan University ²Hitotsubashi University

k.heesu@skku.edu

{shota,komachi}@scl.sds.hit-u.ac.jp

Abstract

We study how a small machine translation model should allocate a fixed data budget between parametric training and retrieval memory. We define a shared 100M-token corpus-access ledger that charges pretraining, fine-tuning, and retrieval-datastore tokens alike. Using $\approx 42\text{M}$ -parameter models, we compare three uses of the same per-pair datastore across 12 directed pairs of English, Chinese, Japanese, and Korean: leaving it unused, training on it, or retrieving from it into the prompt. We evaluate retrieval additivity across all 12 pairs and analyze retrieval-conditioned behavior in a four-pair core set. Retrieval shows no consistent additive gain over simply not using the datastore, a result that persists under an off-budget dense retriever and pools up to approximately $1000\times$ larger. On the four-pair core set, retrieval-trained models are nevertheless highly context-dependent: removing retrieved examples causes large score drops, while replacing them with query-unrelated, count-matched same-pair examples recovers nearly the full score. This suggests that the learned dependence is primarily on exemplar-shaped context rather than on query-specific information carried by the retrieved neighbors. Our training and evaluation code is available at <https://github.com/kheesu/retrieval-without-relevance>.

1 Introduction

Retrieval-augmented systems are often evaluated by holding a model’s training data fixed while granting access to an additional datastore. Such comparisons show whether retrieval helps with extra corpus access, but not whether the same data would be more useful if incorporated into the model’s parameters. Prompt-level retrieval introduces a second attribution problem: any observed effect may arise from the information in the selected neighbors or from the model’s dependence

The model needs an example, not the nearest neighbor

Same retrieval-trained model; only the prompt context changes
Chinese→English · 100k fine-tuning pairs

SOURCE (ZH)

这为我们建立仿真模型提供了大量的信息和材料，有助于我们理解大脑运作的过程。

REFERENCE

This provides us with a lot of information and material to build simulation models which help us to understand processes in our mind.

NEAREST RETRIEVED EXAMPLE (1 OF 3 IN PROMPT)

“We believe that it provides the basis for a good compromise.”

No datastore $\approx$ retrieved $\approx$ random $\gg$ example removed

NO DATASTORE (S-only)

separately trained, no retrieval
This provides a great deal of information and materials to help us understand the processes of the brain.

RETRIEVED EXAMPLE (Retrieve-D) *nearest example in prompt*

This provides a large number of information and materials to help us understand the processes of the brain.

RANDOM EXAMPLE (Random-context)

unrelated, same slot
same as Retrieved example except “number” $\rightarrow$ “amount”

EXAMPLE REMOVED (No-context)

context slot deleted
That is, for example, for instance, in the context of the biennium, and for example in the biennial process.

CORPUS chrF (FLORES test, 3 seeds)

No datastore 34.8 · Retrieved example 34.4

Random example 34.4 · Example removed 18.1

Figure 1: **Query-matched neighbor identity is unnecessary in this example, but exemplar-shaped context is required.** For a held-out FLORES zh→en example, the retrieval-trained RETRIEVE-D model translates correctly whether conditioned on its nearest datastore match or on a query-unrelated, count-matched same-pair example, but degenerates once the demonstration slot is removed. An independently trained S-ONLY model matches these outputs without ever seeing the datastore. Selection protocol in Appendix H.

on the presence of exemplar-formatted context. It therefore remains unclear whether retrieval functions as useful external memory or merely changes the prompt interface that the model learns to expect.

We study these questions in machine transla-

tion, where parallel sentence pairs can be used directly either for supervised training or as retrieved prompt context. Using $\approx 42\text{M}$ -parameter decoder-only models, we evaluate all 12 directed pairs among English, Chinese, Japanese, and Korean under a shared 100M-token corpus-access budget covering pretraining, fine-tuning, and datastore tokens. The budget is the BabyLM Challenge’s 100M-word data limit (Warstadt et al., 2023), extended so that a retrieval datastore is charged like any other corpus (Section 3.2); we ask the challenge’s question—what a fixed, small corpus is best spent on—for a system that can hold data outside its weights. For each pair, the same datastore is either left unused, incorporated through supervised training, or retrieved into the prompt. To distinguish neighbor information from interface dependence, we additionally evaluate the retrieval-trained checkpoints after removing the retrieved context or replacing it with count-matched random examples.

This evaluation yields three main findings. First, prompt-level retrieval exceeds the no-datastore control by at most +0.5 chrF across 36 pair-budget cells and is non-positive in 30. This pattern persists with a dense retriever and retrieval pools approximately three orders of magnitude larger. Training on the datastore, by contrast, usually performs best in the most data-scarce regime, improving over the no-datastore control by +5.8 to +18.1 chrF on 10 of 12 pairs at the 1k-pair budget, with gains diminishing as supervised data increases.

Second, on the four-pair core set where we ran inference-time probes, retrieval-trained models nevertheless depend strongly on their prompt interface: removing context causes losses of up to 16.4 chrF, whereas query-unrelated same-pair examples—count-matched on all four pairs and exactly length-matched on $\text{en} \leftrightarrow \text{zh}$ —recover nearly the full score. The models therefore learn to require exemplar-shaped context rather than to use relevant neighbors.

Third, whenever retrieval outperforms training on the datastore, the latter falls below the no-datastore control, showing how baseline degradation can be mistaken for retrieval value. Our claims are limited to prompt-level retrieval in small MT models trained from scratch.

Our contributions are:

- a shared corpus-access ledger and no-datastore control for comparing training and retrieval under a fixed data budget;

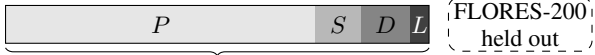
- evidence across 12 directed translation pairs that prompt-level retrieval shows no consistent gain over leaving the datastore unused, whereas incorporating the datastore through fine-tuning primarily helps when supervision is scarce;
- random-context controls on the four-pair core set, plus an exact length-matched control on $\text{en} \leftrightarrow \text{zh}$, showing strong dependence on retrieval-shaped context but weak dependence on query-relevant neighbors; and
- an artifact-qualified attribution analysis illustrating how a degraded datastore-trained baseline can produce an apparent retrieval gain.

2 Related Work

Retrieval as external memory. kNN-LM (Khandelwal et al., 2020) and kNN-MT (Khandelwal et al., 2021) interpolate a parametric model with a nearest-neighbor datastore at the token level; RAG (Lewis et al., 2020) and REALM (Guu et al., 2020) retrieve passages into a generation or masked-LM objective; RETRO (Borgeaud et al., 2022) conditions a language model on retrieved chunks during pretraining; and MassiveDS (Shao et al., 2024) treats datastore scale itself as an axis of improvement. These results are typically reported without charging the datastore against the model’s training data budget, so a gain from retrieval and a gain from more data are not distinguished. We adopt the substitution framing but make the accounting explicit: under a fixed corpus budget, datastore tokens are charged to the same ledger as training tokens, so retrieval competes with training for data rather than adding to it.

Retrieval-augmented machine translation. Translation memory and example retrieval have a long history in MT: neural systems have retrieved and fused matched translation-memory segments (He et al., 2021), retrieved monolingual target-side memory (Cai et al., 2021), and used contrastive retrieval to select memory examples (Cheng et al., 2022). Hao et al. (2023) show that naively retrieved translation examples can backfire in low-resource settings, motivating our no-datastore and random-content controls rather than an assumption that retrieval is neutral-or-better by default. Dictionary-augmented NMT (Ghazvininejad et al., 2023) targets rare or unseen words specifically; we adapt this as a secondary, more narrowly scoped arm.

Random demonstrations as a control. Min

			
one 100M-token ledger per directed pair			
<i>P</i> : Wikipedia pretraining (78–84M tokens)			
<i>S</i> : fine-tuning pairs, 1k $\subset$ 10k $\subset$ 100k (+1k val.)			
<i>D</i> : datastore, disjoint from <i>S</i> , capped to remainder			
<i>L</i> : dictionary (secondary arm, App. F)			

Condition	<i>S</i>	<i>D</i>	Context
S-ONLY	train	—	none
TRAIN-ON-D	train	train	none
RETRIEVE-D	train	prompt	top-3 retrieved
NO-CONTEXT	RETRIEVE-D ckpt	none	none
RANDOM-CONTEXT	RETRIEVE-D ckpt	3 random same-pair	none

Figure 2: **Data pools and conditions.** Top: resources accessible to each condition are charged against a single 100M-token ledger; the diagram shows the pretraining pool *P*, fine-tuning pool *S*, datastore *D*, and (dictionary arm only) lexicon *L*. Bottom: how each condition uses *S* and *D*. All conditions share *P* through one pretrained base per pair. NO-CONTEXT and RANDOM-CONTEXT are inference-only probes on the trained RETRIEVE-D checkpoint.

et al. (2022) show that in-context learning is often insensitive to the correctness of demonstration labels: replacing gold labels with random ones barely hurts few-shot performance, implicating the format, input distribution, and label space of the demonstrations rather than their content. This complements work that instead retrieves or selects demonstrations to improve in-context learning (Liu et al., 2022; Rubin et al., 2022); our random-context control asks the analogous relevance question for neighbor identity in a retrieval-conditioned regime and finds the same qualitative answer.

3 Evaluation Framework

3.1 Allocation Systems and Contrasts

Each directed pair has three disjoint data pools, all charged to one ledger (Figure 2): a pretraining pool *P*, a parallel fine-tuning pool *S*, and a parallel datastore *D*. We compare three uses of *D*: leaving it unused (S-ONLY), incorporating it into supervised fine-tuning (TRAIN-ON-D), or using it for prompt-level retrieval during fine-tuning and inference (RETRIEVE-D). The three trained systems share the same architecture, tokenizer, and frozen decoding configuration (Section 4), but differ in how they use the datastore. NO-CONTEXT and RANDOM-CONTEXT reuse the trained RETRIEVE-D checkpoint and modify only its evaluation-time context.

RETRIEVE-D – S-ONLY measures the additive value of making the datastore available through retrieval. TRAIN-ON-D – S-ONLY measures the effect of incorporating it into supervised fine-tuning, while TRAIN-ON-D – RETRIEVE-D is the direct allocation contrast. NO-CONTEXT measures dependence on the trained prompt interface, whereas RANDOM-CONTEXT tests whether that dependence requires query-relevant neighbors. These contrasts assume a shared accounting of data, which we define in the next section.

3.2 Shared data ledger

All model-facing data are charged to one ledger: pretraining tokens, supervised fine-tuning tokens, datastore tokens, and dictionary tokens.

$$B_c = N(P) + N(S_c) + I_c^D N(D) + I_c^L N(L) \leq 100M,$$

where $N(X)$ is the number of BPE token occurrences in the deduplicated records of resource *X*, counted over both source and target sides; *P*, *S_c*, *D*, and *L* are the pretraining, fine-tuning, datastore, and dictionary resources; and I_c^D , I_c^L indicate whether condition *c* has access to *D*, *L* at all. This ledger is our extension of the BabyLM data budget (Warstadt et al., 2023) to systems that hold data outside the weights; its relation to the official word-based rule is discussed in the Limitations section. Token occurrences are charged once per accessible resource per condition, so an epoch-trained model and a repeatedly queried datastore are comparable. This makes retrieval an allocation decision rather than an extra-data decision: the same datastore tokens can be left unused, used for training, or held in a lookup table. We keep these uses mutually exclusive across our five conditions so that no condition both trains on *D* and retrieves from it. The ledger’s own accounting would not forbid a combined condition, and we leave that hybrid to future work.

3.3 Retrieval and prompting

RETRIEVE-D’s retriever uses TF-IDF over the datastore’s source side, with language-aware tokenization for all languages, shared with the dictionary arm of Appendix F (kiwipiepy using the default model for Korean, jieba with its default dictionary for Chinese, and fugashi with the unidic-lite dictionary for Japanese), and prepends the top-*k*=3 retrieved examples to the source sentence before a translation marker and the target; the fine-tuning

loss is computed on target tokens only, so retrieval shapes conditioning without itself being a prediction target. This lexical retriever is budget-clean by construction; we additionally report a LaBSE (Feng et al., 2022) dense retriever as an off-budget upper-reference and find it indistinguishable from the lexical retriever (Section 5.3). The rationale for treating LaBSE as off-budget and the prompt template are given in Appendix C.

4 Experimental Setup

Languages and pairs. We study English (en), Chinese (zh), Japanese (ja), and Korean (ko), evaluating directed translation across all 12 directed pairs: both directions of $\text{en} \leftrightarrow \text{zh}$, $\text{en} \leftrightarrow \text{ja}$, $\text{en} \leftrightarrow \text{ko}$, $\text{zh} \leftrightarrow \text{ja}$, $\text{zh} \leftrightarrow \text{ko}$, and $\text{ja} \leftrightarrow \text{ko}$. A core set of four pairs ($\text{en} \leftrightarrow \text{zh}$, $\text{ko} \leftrightarrow \text{zh}$) receives the full 3-seed design (S-ONLY, TRAIN-ON-D, RETRIEVE-D, plus the two inference-only probes) and anchors the main results; the S-ONLY control is extended to the remaining 8 pairs to test generalization (Section 5.1); and a breadth grid crosses TRAIN-ON-D, a secondary dictionary-retrieval arm (DICT-RETRIEVE; Appendix F), and a LaBSE-retriever RETRIEVE-D over all 12 pairs for external validity (Appendix B). Chinese is normalized to Simplified throughout. Cross-pair comparisons are descriptive only, since data sources and domains differ by pair.

Data. Each directed model is charged for pre-training, fine-tuning, and datastore tokens under the 100M-token ledger of Section 3.2. Pretraining uses language-specific Wikipedia. Parallel data are pair-specific: the Korean-pivot CJK pairs ($\text{ko} \leftrightarrow \text{zh}$, $\text{ko} \leftrightarrow \text{ja}$) use AI-Hub corpora; $\text{en} \leftrightarrow \text{ko}$ combines AI-Hub with WikiMatrix (Schwenk et al., 2021); $\text{en} \leftrightarrow \text{zh}$ combines MultiUN (Eisele and Chen, 2010), WikiMatrix, and Tatoeba; and $\text{en} \leftrightarrow \text{ja}$ and $\text{zh} \leftrightarrow \text{ja}$ use WikiMatrix alone. Within each pair, fine-tuning examples and the retrieval datastore are disjoint; for the Korean-pivot CJK pairs they additionally come from different registers of the same corpus (literary books for fine-tuning, broadcast captions for the datastore), a mismatch Section 5.4 returns to. The test set is FLORES-200 devtest (NLLB Team et al., 2022) (1,012 sentences per pair), held out and not counted against the budget; a confirmatory re-evaluation additionally uses FLORES-200 dev (997 sentences per pair), likewise held out and off-budget (Section 5.2, Appendix E). Datastores are token-capped per pair so that every trained system remains within budget.

Full provenance, licenses, exact source mixtures and counts, per-pair budget accounting, and the contamination audit are in Appendix A.

Model. Every condition and pair uses the same architecture: a GPT-2-style (Radford et al., 2019) decoder-only Transformer, 8 layers, 512 hidden, $\approx 42\text{M}$ parameters, trained from scratch, with a corpus-driven 32k-vocabulary BPE (Sennrich et al., 2016) tokenizer trained on the pretraining corpus. Each translation pair is fine-tuned from a pretrained base whose BPE covers the pair’s scripts; all conditions for a directed pair share the same base, so any difference between conditions is attributable to how the datastore is used rather than to architecture or tokenization. Pretraining is capped at 78–84M tokens per base so that pretraining, fine-tuning, and datastore tokens jointly fit under the 100M-token ledger (Appendix A).

Training and decoding. Supervised fine-tuning budgets are nested, $1\text{k} \subset 10\text{k} \subset 100\text{k}$ pairs, with a shared held-out validation split. Training uses 3 fine-tuning seeds per core-set cell, and the breadth grid uses 3 seeds throughout; in the extended no-datastore control (Section 5.1) the TRAIN-ON-D and RETRIEVE-D arms use 3 seeds, the S-ONLY control uses 3 seeds on a subset of borderline cells and a single seed elsewhere (Appendix C), and other descriptive sweeps are single-seed where noted. All conditions, budgets, and seeds share one frozen decoding configuration (beam search, beam width 4), so decoding differences can never confound a cross-condition comparison. Because TRAIN-ON-D runs the same number of epochs over $S \cup D$, it receives more optimizer updates and cumulative token exposure than the other trained conditions. All seed variation is in supervised fine-tuning; every condition for a given pair shares one pretraining run, so cross-seed confidence intervals capture fine-tuning randomness and understate total run-to-run variance, and results are conditional on one pretrained base per pair (see Limitations). Optimizer settings, checkpoint selection, beam-search details, the seed-allocation rule for the extended S-ONLY control, and a repetition-control decision made before the evaluation grid ran are given in Appendix C.

Metrics and significance. The headline metric is plain chrF (Popović, 2015) (sacreBLEU (Post, 2018) CHRF, word_order=0); BLEU (Papineni et al., 2002), TER (Snover et al., 2006), COMET (Rei et al., 2020), and chrF++ (Popović, 2017) are secondary and reported in Appendix D,

pair	budget	S-only	Train-D	Retr-D	Rand-ctx	No-ctx	Δ Train	Δ Retr
zh-en	1k	17.9	36.0	16.7	17.3	15.6	+18.1	-1.2
zh-en	10k	26.9	35.8	25.4	24.9	16.9	+8.9	-1.5
zh-en	100k	34.8	36.6	34.4	34.4	18.1	+1.9	-0.3
en-zh	1k	3.8	16.9	3.6	2.5	2.2	+13.1	-0.2
en-zh	10k	9.9	16.9	9.1	8.4	4.4	+7.0	-0.8
en-zh	100k	15.9	17.7	15.8	15.5	5.0	+1.8	-0.2
ko-zh	1k	5.0	3.7	4.6	4.3	3.3	-1.3	-0.4
ko-zh	10k	10.0	8.3	9.8	9.7	4.9	-1.7	-0.2
ko-zh	100k	14.4	13.5	14.3	14.3	5.1	-0.9	-0.2
zh-ko	1k	6.0	5.4	6.0	5.7	4.4	-0.6	0.0
zh-ko	10k	11.3	6.4	11.0	11.0	6.4	-4.9	-0.3
zh-ko	100k	15.1	14.1	14.9	14.8	4.1	-1.0	-0.2

Table 1: Core-set attribution results (3-seed means, plain chrF). S-only = S-ONLY, Train-D = TRAIN-ON-D, Retr-D = RETRIEVE-D, Rand-ctx = RANDOM-CONTEXT, No-ctx = NO-CONTEXT (Figure 2). Δ Train = Train-D - S-only; Δ Retr = Retr-D - S-only. Deltas are computed from unrounded seed means. ko $\leftrightarrow$ zh TRAIN-ON-D values use the original datastore; values after removing its markup artifact are in Appendix G. All twelve Δ Train 3-seed Student-*t* 95% CIs exclude zero; six of twelve Δ Retr CIs do, all negative (Appendix D).

where the core-set results are also confirmed metric-robust. Primary contrasts are within-pair paired deltas on the same FLORES sources, aggregated across seeds with Student-*t* 95% confidence intervals, with false-discovery-rate control across the family of per-seed tests (Appendix D).

The dictionary-retrieval arm is confined to the appendices (Appendix F; Table 7) because coverage collapsed with budget and a tokenizer mismatch prevented a clean retrieval-quality interpretation.

5 Results

Unless otherwise stated, all scores are plain chrF on FLORES-200 devtest ($n = 1,012$), decoded with the frozen configuration of Section 4, and reported as 3-seed means; single-seed cells are marked explicitly.

5.1 Retrieval adds no value over the no-datastore control

Table 1 compares the no-datastore control (S-ONLY) with TRAIN-ON-D (datastore baked into weights) and RETRIEVE-D (datastore retrieved, budget-clean lexical retriever), together with the two inference-only probes.

Retrieval does not beat the no-datastore control on the core set. Δ RETRIEVE-D lies between -1.5 and 0.0 chrF in every core-set cell. This holds for the budget-clean lexical retriever, for the LaBSE upper-reference, and under every secondary metric (Appendix D). Holding the datastore in a retriever is at best neutral, and typically slightly negative, relative to not having the datastore at all.

TRAIN-ON-D gains diminish with additional supervision. On en $\leftrightarrow$ zh, Δ TRAIN-ON-D is +13 to +18 chrF at 1k and shrinks monotonically to +1.8 and +1.9 at 100k, never turning negative. ko $\leftrightarrow$ zh is the outlier: Δ TRAIN-ON-D is negative at every budget (-0.6 to -1.7), with one deeper, seed-unstable cell (zh-ko 10k, -4.9) partly traced to a datastore artifact in Appendix G.

The null generalizes to all 12 pairs. Extending the S-ONLY control to the remaining 8 pairs (Table 2) yields Δ RETRIEVE-D $\leq$ +0.5 chrF in all 24 new cells; 21 of 24 are $\leq$ +0.1. Across all 36 pair-budget cells, 30 deltas are non-positive and 34 are at most +0.2. In these 24 cells RETRIEVE-D is the LaBSE variant from the breadth grid, since the lexical retriever is instantiated only on the core set; the extension therefore tests the qualitative pattern rather than the budget-clean accounting. Where both retrievers were run, they differ by at most 0.1 chrF (Section 5.3). The largest positive delta (en-ja at 1k, +0.5) survives a 3-seed re-run (S-ONLY range across seeds 0.11; per-seed Δ +0.42 to +0.50) but sits at an absolute chrF of $\approx$ 5, well below usable translation quality. Nowhere does retrieval beat the no-datastore control at a budget where the systems are usable.

5.2 Context is required; retrieval relevance is not

The analysis in this section uses only the four core pairs, on which the inference-time probes were run; the eight-pair extension of Section 5.1 tests the null, not the mechanism.

pair	budget	Δ TRAIN-ON-D	Δ RETRIEVE-D
en-ja	1k/10k/100k	+12.4 / +7.6 / +2.0	+0.5 / -0.9 / -0.3
ja-en	1k/10k/100k	+15.7 / +9.2 / +2.1	-0.1 / -1.6 / -0.6
en-ko	1k/10k/100k	+10.3 / +6.4 / +1.7	+0.1 / -0.7 / -0.1
ko-en	1k/10k/100k	+16.6 / +8.9 / +2.2	+0.3 / -1.7 / -0.7
ja-ko	1k/10k/100k	+9.8 / +3.3 / +1.0	-0.4 / -0.3 / -0.3
ko-ja	1k/10k/100k	+8.0 / +2.1 / +0.4	-0.7 / -0.3 / 0.0
ja-zh	1k/10k/100k	+5.8 / +3.4 / +1.0	-0.4 / +0.1 / +0.1
zh-ja	1k/10k/100k	+6.9 / +4.9 / +0.7	-0.3 / 0.0 / +0.2

Table 2: No-datastore control extended to the remaining 8 pairs, all vs. S-ONLY (plain chrF). TRAIN-ON-D and RETRIEVE-D are 3-seed means; S-ONLY is 3-seed on the 10 borderline cells and single-seed elsewhere (Appendix C). RETRIEVE-D here is the LaBSE-retriever variant (Section 5.3). With Table 1 this covers all 12 directed pairs.

Removing context collapses performance.

Stripping retrieved context from the trained RETRIEVE-D checkpoints (NO-CONTEXT, same frozen decoding) costs 16.4 chrF at zh-en 100k ($34.4 \rightarrow 18.1$), 10.8 at en-zh 100k, 9.2 at ko-zh 100k, and 10.8 at zh-ko 100k (Table 1). The model is strongly conditioned on retrieval-shaped input; it is not ignoring the datastore.

Replacing query-matched neighbors with random same-pair examples costs almost nothing.

RANDOM-CONTEXT feeds the same checkpoints count-matched random datastore pairs and recovers essentially the full RETRIEVE-D level at 10k and 100k: within 0.3 chrF on every 100k cell and within 0.7 at 10k. At 1k the gaps are larger relative to the low absolute scores but mixed in sign: -1.1 on en-zh, +0.6 on zh-en. The dependency decomposes cleanly:

$$\text{S-ONLY} \approx \text{RETRIEVE-D} \approx \text{RANDOM-CONTEXT} \\ \gg \text{NO-CONTEXT}.$$

The model therefore depends strongly on the retrieval-trained exemplar format, while the identity of the query-matched neighbor contributes little. Importantly, RANDOM-CONTEXT draws examples from the same directed language pair: it breaks query relevance while preserving language-pair identity and parallel-example structure. We therefore interpret this control as evidence against strong dependence on nearest-neighbor relevance, not as evidence that arbitrary context would suffice. “Exemplar-shaped context” should accordingly be read as the retrieval-time format together with whatever pair-level information same-pair exemplars carry; separating the two would require cross-pair or non-parallel contexts, which we did not run (see

Limitations). For a model fine-tuned into the dependency, this parallels the random-demonstration result of Min et al. (2022). The ordering also holds under BLEU, TER, and COMET (Appendix D); the headline is not a chrF artifact.

The result survives exact length matching.

RANDOM-CONTEXT matches the *number* of retrieved exemplars but not their length, so we confirm the result with a length-matched control (LM-RANDOM): in a confirmatory re-evaluation on FLORES-200 dev, with the same checkpoints and frozen decoding, the three retrieved exemplars are replaced by random datastore pairs whose rendered BPE length exactly matches the retrieved ones’ (Appendix E). The 100k ordering replicates on this confirmatory split: RETRIEVE-D-S-ONLY stays within $[-0.19, +0.13]$ chrF on all four core-set pairs, RANDOM-CONTEXT stays within -0.2 of RETRIEVE-D, and NO-CONTEXT again collapses (Table 8, Appendix E). At 100k, LM-RANDOM sits only 0.14 (en $\rightarrow$ zh) and 0.25 (zh $\rightarrow$ en) chrF below RETRIEVE-D, against losses of 10.9–16.9 chrF on these two pairs when context is removed: most retrieval-conditioned performance does not depend on neighbor identity even with exemplar length held fixed. On zh $\rightarrow$ ko dev, a sizable fraction of stripped-context outputs are empty strings rather than fluent-but-wrong translations (Appendix E); NO-CONTEXT reflects dependence on the retrieval-conditioned interface, not productive use of retrieved information.

5.3 Robustness: retriever quality and datastore size

Two potential confounds could explain the null of Section 5.1: a retriever that is too weak, or a datastore that is too small.

Retriever quality. Table 3 compares the budget-clean lexical retriever against the off-budget LaBSE-dense reference on five cells. Differences range from -0.07 to +0.09 chrF, within cross-seed variation. Two consequences follow: the budget-clean retriever gives up nothing measurable relative to the off-budget one, so no accounting exception for a pretrained retriever is needed; and the non-additivity of Section 5.1 is not attributable to retriever weakness in these cells. This is consistent with the null being about neighbor relevance rather than retriever strength (Section 5.2), though it does not bound an oracle or task-trained retriever.

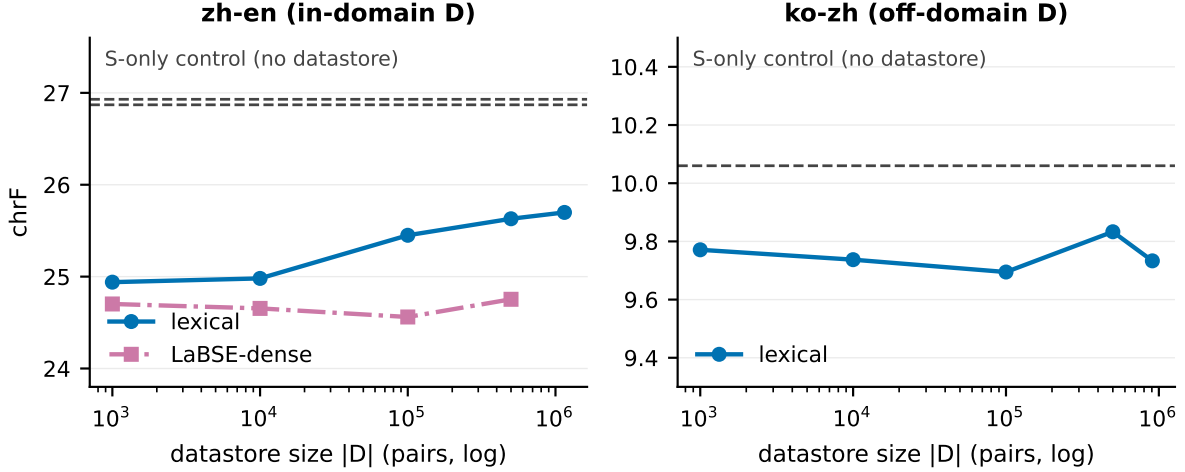


Figure 3: RETRIEVE-D chrF vs. datastore size $|D|$ (log scale, single seed per point), fixed $S = 10k$, against the S-ONLY control. Every curve stays below its control level across a $\sim 1,000\times$ range of $|D|$; only the in-domain zh-en lexical sweep shows a mild, plateauing slope.

pair	budget	lexical	LaBSE	lex.—LaBSE
zh-ko	10k	10.97	11.02	−0.05
zh-ko	100k	14.90	14.98	−0.07
ko-zh	100k	14.25	14.25	0.00
zh-en	100k	34.44	34.46	−0.02
en-zh	100k	15.75	15.66	+0.09

Table 3: RETRIEVE-D with the budget-clean lexical retriever vs. the off-budget LaBSE-dense upper-reference (3-seed, plain chrF). Two decimals are reported because the contrasts are below 0.1 chrF.

Datastore size. Table 4 and Figure 3 report inference-only sweeps on fixed RETRIEVE-D checkpoints ($S = 10k$), growing the datastore from 1k pairs to the full disjoint pool (off-budget by construction). We sweep both retrievers on zh-en, whose datastore is partly Wikipedia-derived (WikiMatrix) and therefore closest in domain to FLORES, and the lexical retriever on ko-zh, whose broadcast-caption datastore is off-domain. Every curve sits below the no-datastore control at every size. The only monotone gain is the zh-en lexical sweep (+0.76 chrF over a $1,150\times$ pool increase, plateauing); the dense retriever is flat (± 0.2 , non-monotone), and the ko-zh sweep is flat (± 0.1 over $\sim 900\times$). Even the largest tested datastore does not reach the no-datastore control. This is unsurprising given Section 5.2: if neighbor identity barely matters, better neighbors drawn from a larger pool cannot help.

5.4 Apparent retrieval wins decompose into baseline harm

The budget-matched allocation contrast, TRAIN-ON-D vs. RETRIEVE-D, is misleading without the no-datastore control. Table 5 lists every one of the 36 pair-budget cells in which RETRIEVE-D significantly beats TRAIN-ON-D (3-seed CIs exclude zero): the six $ko \leftrightarrow zh$ cells. The contrast is significantly negative in the remaining 30. All six decompose the same way: RETRIEVE-D $\approx$ S-ONLY (retrieval adds nothing), while TRAIN-ON-D falls below the control level (training on the datastore actively hurts). The apparent retrieval gain is therefore an avoided baseline loss, not added retrieval value: “retrieval beats weights” here means that retrieval avoids a harm that baking-in caused, and for these cells the best allocation is to spend no budget on the datastore at all.

The general form is that when a baseline’s treatment can be harmful, a comparison against that baseline conflates the competitor’s gain with the baseline’s loss. Confidence intervals that exclude zero do not rule this out: significance validates the contrast that was run, not the attribution inferred from it.

Domain match does not explain the sign pattern.

Across all 12 pairs at the 1k budget, Δ TRAIN-ON-D is positive on 10 of 12 (+5.8 to +18.1); only $ko \leftrightarrow zh$ is negative. This invites a domain reading: $en \leftrightarrow zh$ ’s WikiMatrix/MultiUN datastore is in-domain for FLORES and gains, while $ko \leftrightarrow zh$ ’s AI-Hub captions are off-domain and lose. We spec-

sweep	$ D $ 1k	10k	100k	500k	full pool	S-ONLY control
zh-en, lexical	24.94	24.98	25.45	25.63	25.70 (1.15M)	26.93
zh-en, LaBSE-dense	24.70	24.65	24.56	24.75	—	26.87
ko-zh, lexical	9.77	9.74	9.69	9.83	9.73 (907k)	10.06

Table 4: Datastore-size sweep on fixed RETRIEVE-D checkpoints ($S = 10k$, plain chrF). Each sweep uses a single fine-tuning seed and the S-ONLY column reports that seed’s control (seed 1 for the two lexical sweeps, seed 2 for the LaBSE sweep), which is why the two zh-en rows show different control values. The LaBSE full-pool point was not run (—).

cell	Retr-Train	95% CI	Retr-S	Train-S	reading
ko-zh 1k	+0.91	[0.28, 1.53]	−0.4	−1.3	baseline harm
ko-zh 10k	+1.49	[1.13, 1.86]	−0.2	−1.7	baseline harm
ko-zh 100k	+0.73	[0.39, 1.06]	−0.2	−0.9	baseline harm
zh-ko 1k	+0.60	[0.18, 1.02]	0.0	−0.6	baseline harm
zh-ko 10k	+4.60	[3.76, 5.45]	−0.3	−4.9	baseline harm
zh-ko 100k	+0.81	[0.44, 1.18]	−0.2	−1.0	baseline harm

Table 5: Decomposing every positive Retrieve-D vs. Train-D delta among all 12 pairs (paired 3-seed Student- t 95% CIs; all six CIs exclude zero). The six positive cells are core-set cells with the lexical retriever; the contrast is significantly negative in the remaining 30 cells, which include the LaBSE-retriever extension (Table 2). Retr = RETRIEVE-D, Train = TRAIN-ON-D, S = S-ONLY.

ified domain predictions for the four undirected extension pairs before examining their results (positive for $\text{en} \leftrightarrow \text{ja}$ and $\text{zh} \leftrightarrow \text{ja}$, negative for $\text{ja} \leftrightarrow \text{ko}$, borderline for $\text{en} \leftrightarrow \text{ko}$; Appendix B), and the prediction fails: $\text{ja} \leftrightarrow \text{ko}$, trained against the same off-domain caption store as $\text{ko} \leftrightarrow \text{zh}$, gains $+9.8/+8.0$ chrF at 1k, the opposite of the predicted sign. The full grid instead supports a budget-gated boost from extra bitext of nearly any register—large when data-starved, fading to $+0.4$ to $+2.2$ by 100k without flipping sign—with $\text{ko} \leftrightarrow \text{zh}$ a pair-level outlier (Appendix G), not evidence of a domain law. Full predictions and outcomes are given in Appendix B.

Caveat. All six $\text{ko} \leftrightarrow \text{zh}$ cells in Table 5 use RETRIEVE-D checkpoints trained and evaluated with artifact-bearing datastore context; only TRAIN-ON-D was rerun after cleaning (Appendix G), which isolates the weights-memory pathway but not the conditioning pathway. The risk concentrates in zh-ko 10k, where cleaning shifted TRAIN-ON-D by $+2.2$ chrF (3-seed mean) and altered seed-level stability; the other cells shift by at most 0.3 chrF. The decomposition is therefore diagnostic rather than fully clean causal evidence for $\text{ko} \leftrightarrow \text{zh}$, but the central RETRIEVE-D-vs.-S-ONLY result of Section 5.1 is unaffected.

6 Discussion

Dependency without value. The substitution thesis (Borgeaud et al., 2022; Khandelwal et al., 2021; Shao et al., 2024) predicts that datastore tokens can do the work of tokens trained into weights. Under our shared-ledger accounting, the prediction fails in a specific way: retrieval provides no consistent additive gain across the 12 pairs (Section 5.1), yet on the four core pairs where we probed it, it is not ignored either (Section 5.2). Retrieval training creates an input dependency whose system-level payoff is zero: the model learns to need the interface and, with it, recovers only the level of a model trained never to need it. “Small models ignore retrieval” is the wrong summary; retrieval induces dependency without value.

Relation to BabyLM. We make no cognitive claims. Our question is the BabyLM question posed at the systems level: given a corpus on the scale the challenge treats as developmentally plausible, is that corpus better spent internalized in the weights or kept as consultable memory? For translation with $\approx 42M$ parameters, the answer is that consultable memory is not used as memory: the model internalizes the retrieval interface but shows little dependence on the query-specific content of the retrieved neighbors, and the data would have been more useful in the weights whenever it was useful at all. Two further points connect to the workshop’s aims: the ledger extends BabyLM’s data accounting to systems that hold data outside the weights, and the English–Chinese–Japanese–Korean setting exercises the budget on scripts and language pairs beyond English.

Consequences for evaluation design. A retrieval-MT comparison that reports a retrieval-augmented system against a datastore-trained baseline, without a no-datastore control, cannot distinguish a genuine retrieval gain from an avoided baseline loss (Section 5.4). The risk

is largest exactly where noisy or mismatched datastores and small fine-tuning sets are common. We therefore recommend reporting a no-datastore control alongside any retrieval system evaluated against a datastore-trained baseline.

7 Conclusion

We asked how a small machine translation model should spend a fixed corpus budget when the same parallel data can either be trained into its weights or retrieved into its prompt. Under a shared 100M-token ledger that charges pretraining, fine-tuning, and datastore tokens alike, we compared leaving a pair-specific datastore unused, training on it, and retrieving from it, across 12 directed pairs among English, Chinese, Japanese, and Korean, with ≈ 42 M-parameter models trained from scratch.

Three findings follow. First, prompt-level retrieval never exceeds the no-datastore control by more than +0.5 chrF across 36 pair-budget cells and is non-positive in 30; the null persists under an off-budget dense retriever and retrieval pools up to $\sim 1,000\times$ larger. Second, training on the same datastore is the better use of the budget when supervision is scarce (+5.8 to +18.1 chrF at 1k pairs on 10 of 12 pairs), with gains fading to +0.4–2.2 chrF by 100k. Third, on the four-pair core set, retrieval-trained models are nonetheless strongly dependent on their prompt interface: removing context costs up to 16.4 chrF, while query-unrelated same-pair examples—count-matched on all four pairs and exactly length-matched on $\text{en}\leftrightarrow\text{zh}$ —recover nearly the full score. Retrieval induces dependence without value.

These findings carry a diagnostic consequence: every cell in which retrieval “beats” training on the datastore is one where training fell below the no-datastore control, so the apparent win is an avoided baseline loss rather than added retrieval value. Our claims are limited to prompt-level retrieval in small models trained from scratch and to same-pair relevance-breaking controls. They motivate two reporting practices for retrieval evaluation under constrained corpus access: a no-datastore control that separates retrieval gains from baseline harm, and a relevance-breaking control that separates use of retrieved information from dependence on the retrieval-conditioned interface.

Limitations

Mechanism. Our negative result is scoped to prompt-level (in-context) injection. It does not cover kNN-style distribution interpolation (Khandelwal et al., 2021), whose stronger coupling might extract value where prompting does not.

Controls. RANDOM-CONTEXT and LM-RANDOM draw replacement exemplars from the same directed pair, so they break query relevance while preserving language-pair identity and parallel structure. They rule out dependence on query-specific neighbor information, not on same-pair exemplars as such; contexts drawn from another language pair, from non-parallel text, or with shuffled source–target alignment would be needed to separate format from pair-level content, and we did not run them. LM-RANDOM was run on $\text{en}\leftrightarrow\text{zh}$ at the 100k budget only.

Model, task, and data. All experiments use ≈ 42 M-parameter decoder-only Transformers trained from scratch, evaluated on machine translation prompting with these specific datastores. Larger or differently pretrained models could behave differently.

Seeds. Each pair uses one pretraining seed; all seed variation is in fine-tuning. Cross-seed intervals therefore capture fine-tuning randomness and understate total run-to-run variance.

Accounting. Results are conditional on our corpus-access ledger (Section 3.2), which charges BPE-token occurrences once per accessible resource. It departs from the official BabyLM rule in two ways: we count tokens rather than words, because word boundaries are ill defined for Chinese and Japanese; and we charge fine-tuning and datastore tokens, not only pretraining tokens. Different accounting choices would shift the allocation contrasts.

Excluded arms and open outliers. A secondary dictionary-retrieval arm is disclosed but excluded from retrieval-quality and mechanism conclusions: its coverage collapses with fine-tuning budget, and a tokenizer mismatch confounds the CJK-source evaluations (Appendix F). The $\text{ko}\leftrightarrow\text{zh}$ pair’s residual gap from its control level remains an unexplained pair-level outlier (Appendix G). It does not bear on the central RETRIEVE-D-vs.-S-ONLY result, but the baseline-harm decomposition of Sec-

tion 5.4 is drawn entirely from this pair and is qualified accordingly.

Acknowledgments

This research used datasets from ‘The Open AI Dataset Project (AI-Hub, S. Korea)’; all data information can be accessed through AI-Hub (<https://www.aihub.or.kr>).

References

- Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, 57(1):289–300.
- Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, and 9 others. 2022. [Improving language models by retrieving from trillions of tokens](#). In *International Conference on Machine Learning*, pages 2206–2240. PMLR.
- Deng Cai, Yan Wang, Huayang Li, Wai Lam, and Lemao Liu. 2021. [Neural machine translation with monolingual translation memory](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7307–7318, Online. Association for Computational Linguistics.
- Xin Cheng, Shen Gao, Lemao Liu, Dongyan Zhao, and Rui Yan. 2022. [Neural machine translation with contrastive translation memories](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3591–3601, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Andreas Eisele and Yu Chen. 2010. MultiUN: A multilingual corpus from united nation documents. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC’10)*, pages 2868–2872.
- Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Ariavazhagan, and Wei Wang. 2022. [Language-agnostic BERT sentence embedding](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics.
- Marjan Ghazvininejad, Hila Gonen, and Luke Zettlemoyer. 2023. [Dictionary-based phrase-level prompting of large language models for machine translation](#). *arXiv preprint arXiv:2302.07856*.
- Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. [REALM: Retrieval-augmented language model pre-training](#). In *Proceedings of the 37th International Conference on Machine Learning*. PMLR.
- Hongkun Hao, Guoping Huang, Lemao Liu, Zhirui Zhang, Shuming Shi, and Rui Wang. 2023. [Rethinking translation memory augmented neural machine translation](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2589–2605.
- Qiuxiang He, Guoping Huang, Qu Cui, Li Li, and Lemao Liu. 2021. [Fast and accurate neural machine translation with translation memory](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3170–3180, Online. Association for Computational Linguistics.
- Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. [Nearest neighbor machine translation](#). In *International Conference on Learning Representations*.
- Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. [Generalization through memorization: Nearest neighbor language models](#). In *International Conference on Learning Representations*.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-augmented generation for knowledge-intensive NLP tasks](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.
- Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. [What makes good in-context examples for GPT-3?](#) In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.
- Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.
- Sewon Min, Xinxin Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. [Rethinking the role of demonstrations: What makes in-context learning work?](#) In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064.
- NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Hefernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, and Al Youngblood. 2022. [No language left](#)

- behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. **Bleu: a method for automatic evaluation of machine translation**. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.
- Maja Popović. 2015. **chrF: character n-gram F-score for automatic MT evaluation**. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.
- Maja Popović. 2017. **chrF++: words helping character n-grams**. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.
- Matt Post. 2018. **A call for clarity in reporting BLEU scores**. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. **Language models are unsupervised multitask learners**. *OpenAI blog*, 1(8):9.
- Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. **COMET: A neural framework for MT evaluation**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.
- Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. **Learning to retrieve prompts for in-context learning**. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.
- Holger Schwenk, Vishrav Chaudhary, Shuo Sun, Hongyu Gong, and Francisco Guzmán. 2021. **WikiMatrix: Mining 135M parallel sentences in 1620 language pairs from Wikipedia**. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1351–1361.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. **Neural machine translation of rare words with subword units**. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.
- Rulin Shao, Jacqueline He, Akari Asai, Weijia Shi, Tim Dettmers, Sewon Min, Luke Zettlemoyer, and Pang Wei Koh. 2024. **Scaling retrieval-based language models with a trillion-token datastore**. *arXiv preprint arXiv:2407.12854*.
- Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. **A study of translation edit rate with targeted human annotation**. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.
- Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023. **Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora**. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–34, Singapore. Association for Computational Linguistics.

A Data Sources and Budget Accounting

Pretraining data source. Wikipedia (wikimedia/wikipedia, 20231101 snapshot, CC-BY-SA-4.0), restricted to the pretraining base’s languages and capped to the per-language budget.

Parallel data sources. Each pair has one deduplicated parallel pool, shuffled with a fixed seed and aligned across the two directions, from which the nested fine-tuning splits and the datastore are carved disjointly. The per-pair composition is: ko↔zh and ko↔ja use AI-Hub Korean-pivot MT corpora ($\approx 1.01\text{M}$ and $\approx 1.04\text{M}$ pairs respectively), with the literary-book subcorpus supplying the fine-tuning splits and the broadcast-caption subcorpus supplying the datastore pool; en↔ko merges the AI-Hub book subcorpus (150,543 pairs) with WikiMatrix (Schwenk et al., 2021) (112,124) into a pool of 262,667; en↔zh merges MultiUN (Eisele and Chen, 2010) (capped at 500,000), WikiMatrix (681,382), and Tatoeba (74,606) into a pool of 1,255,636 after cross-source deduplication; en↔ja uses WikiMatrix alone (545,302); and zh↔ja uses WikiMatrix alone (256,579), as constructing zh–ja pairs by pivoting through the Korean- or English-sided corpora proved infeasible (the candidate corpora share almost no pivot-side sentences). WikiMatrix pools use the LASER margin ≥ 1.04 release; the Chinese side of the web-sourced pools is converted to Simplified with OpenCC before splitting.

Dictionary data sources. The dictionary resources are krdict (ko $\leftrightarrow$ zh: 108,686 entries; ko $\leftrightarrow$ ja: 168,532), CC-CEDICT (en $\leftrightarrow$ zh: 152,467), JMdict (en $\leftrightarrow$ ja, capped at 200k), and psyche (the psyche/dictionary_koen Hugging Face dataset, https://huggingface.co/datasets/psyche/dictionary_koen, CC-BY-SA-4.0; derived from the National Institute of Korean Language’s Korean Learners’ Dictionary and Standard Korean Language Dictionary) (en $\leftrightarrow$ ko: 125,906). Reverse directions are constructed by swapping the source and target columns. For zh $\leftrightarrow$ ja, we construct a directional Wiktionary lexicon from Kaikki/Wiktextextract using English sense-aligned translation tables and the Chinese-language section of Japanese Wiktionary (Han-only, OpenCC-normalized; 46,999 entries). Coverage is adequate for zh $\rightarrow$ ja but approximately 0.7% for ja $\rightarrow$ zh. The dictionary analysis therefore covers 10 of the 12 directed pairs; zh $\leftrightarrow$ ja is evaluated only with TRAIN-ON-D and RETRIEVE-D in both directions: the constructed lexicon is usable in one direction only, so we treat zh $\leftrightarrow$ ja as a pair without a usable open-license dictionary and drop the dictionary family (DICT-RETRIEVE, RANDOM-DICT, ORACLE-DICT) for both directions, including zh $\rightarrow$ ja despite its adequate coverage; both directions are marked — in the dictionary columns of Table 7.

Evaluation data. FLORES-200 devtest (NLLB Team et al., 2022) contains 1,012 sentences per pair; directed pairs are constructed from the shared multilingual sentence IDs. FLORES is held out and excluded from the data budget. Exact deduplication and a LaBSE-based semantic audit found no exact overlaps or flagged near-duplicates between FLORES and the training resources.

Budget construction. Each pair uses nested fine-tuning subsets of 1k, 10k, and 100k examples, a 1k validation split, and a pair-specific datastore D . For each directed model, the budget is

$$P + S + D + \text{dict} \leq 100\text{M}$$

when measured with that model’s 32k BPE over both sides of the deduplicated parallel and dictionary records. FLORES is excluded. Pretraining is capped at approximately 78–84M tokens, depending on the pair. The datastore is capped by token count rather than pair count. An initial 200k-pair cap produced totals of 101.8M tokens for en $\leftrightarrow$ zh and 100.7M for en $\leftrightarrow$ ja because these corpora contain longer sentences. The final data construction

sets $|D|$ to $100\text{M} - P - S - \text{dict}$ using a prefix of the shuffled pool shared by both directions of the pair. This yields 10.0M datastore tokens (166,794 pairs; 99.83M total) for en $\leftrightarrow$ zh and 9.5M datastore tokens (182,921 pairs; 99.84M total) for en $\leftrightarrow$ ja; all other pairs were already within budget. This correction affects only TRAIN-ON-D and RETRIEVE-D, which access D ; the dictionary conditions are independent of D .

Pretrained base selection. Each directed pair is fine-tuned from a pretrained base whose tokenizer covers both scripts (e.g., the ko $\leftrightarrow$ zh base for {ko, zh}). Under the fixed pretraining budget, a bilingual base receives approximately 42M tokens per language; trilingual and four-language bases (approximately 28M and 21M tokens per language) were also pretrained but are not used in any reported result; all reported results use the six bilingual bases. Held-out pretraining loss decreased from 7.47 to 3.99 for the ko $\leftrightarrow$ zh base and from 7.33 to 3.81 for the ja $\leftrightarrow$ ko base.

B Full Experimental Grid and Additional Results

Table 6 lists the full experimental grid, including the dictionary family and its inference-only controls, with internal run labels and roles.

Domain-match analysis. We evaluated whether the sign of $\Delta\text{TRAIN-ON-D}$ is explained by domain match between the datastore and FLORES. Before examining the eight-pair extension, we assigned positive effects to the in-domain en $\leftrightarrow$ ja and zh $\leftrightarrow$ ja pairs, negative effects to the off-domain ja $\leftrightarrow$ ko pair, and borderline effects to en $\leftrightarrow$ ko. Table 2 does not support this prediction: ja $\leftrightarrow$ ko, which uses the same off-domain caption datastore as ko $\leftrightarrow$ zh, improves by +9.8/+8.0 chrF at the 1k budget. Across the full grid, TRAIN-ON-D generally provides a large low-budget gain that decreases with additional fine-tuning data. ko $\leftrightarrow$ zh remains the only pair family with consistently negative deltas and is analyzed separately in Appendix G.

Additional 100k-budget results. The breadth grid evaluates {TRAIN-ON-D, DICT-RETRIEVE, RETRIEVE-D-LaBSE} across 12 directed pairs (10 for the dictionary conditions), three fine-tuning budgets, and three seeds; RANDOM-DICT and ORACLE-DICT are added at inference. Because this grid does not contain S-ONLY, it is descriptive with respect to retrieval additivity. Table 7 reports the 100k-budget cells.

Name	Label	Prepended context	Retriever	Role	Train?
S-only	C _S	— none; no datastore at all (fine-tuned on <i>S</i> only)	—	no-datastore control	yes
Train-on-D	C0	— none; datastore baked into weights (fine-tuned on $S \cup D$)	—	weights memory	yes
Retrieve-D	C2	nearest parallel-sentence example(s)	lexical TF-IDF over <i>D</i> ; LaBSE-dense comparison	external phrasal memory	yes
No-context	C2-ret	— context stripped from the trained Retrieve-D model	—	dependence probe	inference
Random-context	C2-rand	count-matched random datastore pairs	random, relevance-broken	neighbor-relevance control	inference
Dict-retrieve	C1	dictionary entries (src $\rightarrow$ tgt) for rare source words	exact lookup + frequency gate	lexical memory (secondary)	yes
Random-dict	C3	random dictionary entries (no source-word overlap)	random, count-matched to Dict-retrieve	prompt-length control	inference
Oracle-dict	C4	dictionary entries whose target appears in the reference	oracle (eval-only)	upper bound for Dict-retrieve	inference

Table 6: Full condition grid, internal labels, and roles. RANDOM-DICT and ORACLE-DICT reuse the DICT-RETRIEVE checkpoint and modify only the inference-time retriever. Reference-dependent oracle retrieval is isolated from all training code (Appendix I).

pair	TRAIN-ON-D	DICT-RETRIEVE	RETRIEVE-D (LaBSE)	RANDOM-DICT	ORACLE-DICT
zh-en	36.6	33.1	34.5	32.1	34.3
ja-en	34.4	31.0	31.7	30.3	32.0
ko-en	35.6	32.6	32.7	31.9	33.2
en-zh	17.7	15.8	15.7	15.8	15.8
en-ja	18.4	16.4	16.1	16.4	15.3
en-ko	16.3	14.5	14.5	14.5	13.7
ko-zh	13.5	13.2	14.3	11.8	13.8
zh-ko	14.1	13.4	15.0	11.5	14.4
ko-ja	23.3	21.6	22.9	19.1	22.3
ja-ko	22.3	14.6	21.0	12.3	18.3
zh-ja	15.5	—	15.0	—	—
ja-zh	13.3	—	12.5	—	—

Table 7: Breadth grid at the 100k budget (3-seed means, plain chrF). The RETRIEVE-D – TRAIN-ON-D confidence interval excludes zero for every pair; the contrast is negative on 10 pairs and positive only on ko $\leftrightarrow$ zh.

C Training, Decoding, and Prompt Construction

Training. Models are pretrained from scratch in bf16 for a fixed number of steps (10 epochs over the packed pretraining corpus at an effective batch of 64 sequences of 512 tokens, i.e. 23,960–25,690 steps depending on the base). Fine-tuning uses 10 epochs, batch size 32 (16 with two gradient-accumulation steps), bf16, dynamic padding, and target-only loss. TRAIN-ON-D is optimized over $S \cup D$, whereas S-ONLY and RETRIEVE-D are optimized over *S*; consequently, TRAIN-ON-D receives more optimizer updates and cumulative token exposure. Optimization uses AdamW (Loshchilov and Hutter, 2019) ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 0.01), learning rate 10^{-4} , cosine decay, 6% warmup, and gradient clipping at 1.0. Checkpoints are evaluated and saved every 200 steps, and the checkpoint with the

lowest validation loss is selected. The learning rate and epoch count were chosen from a small sweep on one TRAIN-ON-D cell and then fixed for all experiments. A 100k-budget S-ONLY fine-tuning run required approximately 56 minutes on an NVIDIA A6000; wall-clock time varies with the training conditions and objectives.

Seed allocation for the extended S-ONLY control. In the 8-pair extension (Appendix B), TRAIN-ON-D and RETRIEVE-D use 3 seeds throughout; the S-ONLY control itself uses 3 seeds on the 10 borderline cells, defined as every extension cell whose initial single-seed Δ RETRIEVE-D was ≥ -0.1 chrF, and a single seed on the remaining 14 cells, where the deltas are far from zero.

Decoding. All systems use beam search with beam width 4, early stopping, length penalty 1.0, no sampling, repetition penalty 1.0, no_repeat_ngram_size=3, and at most 256 new

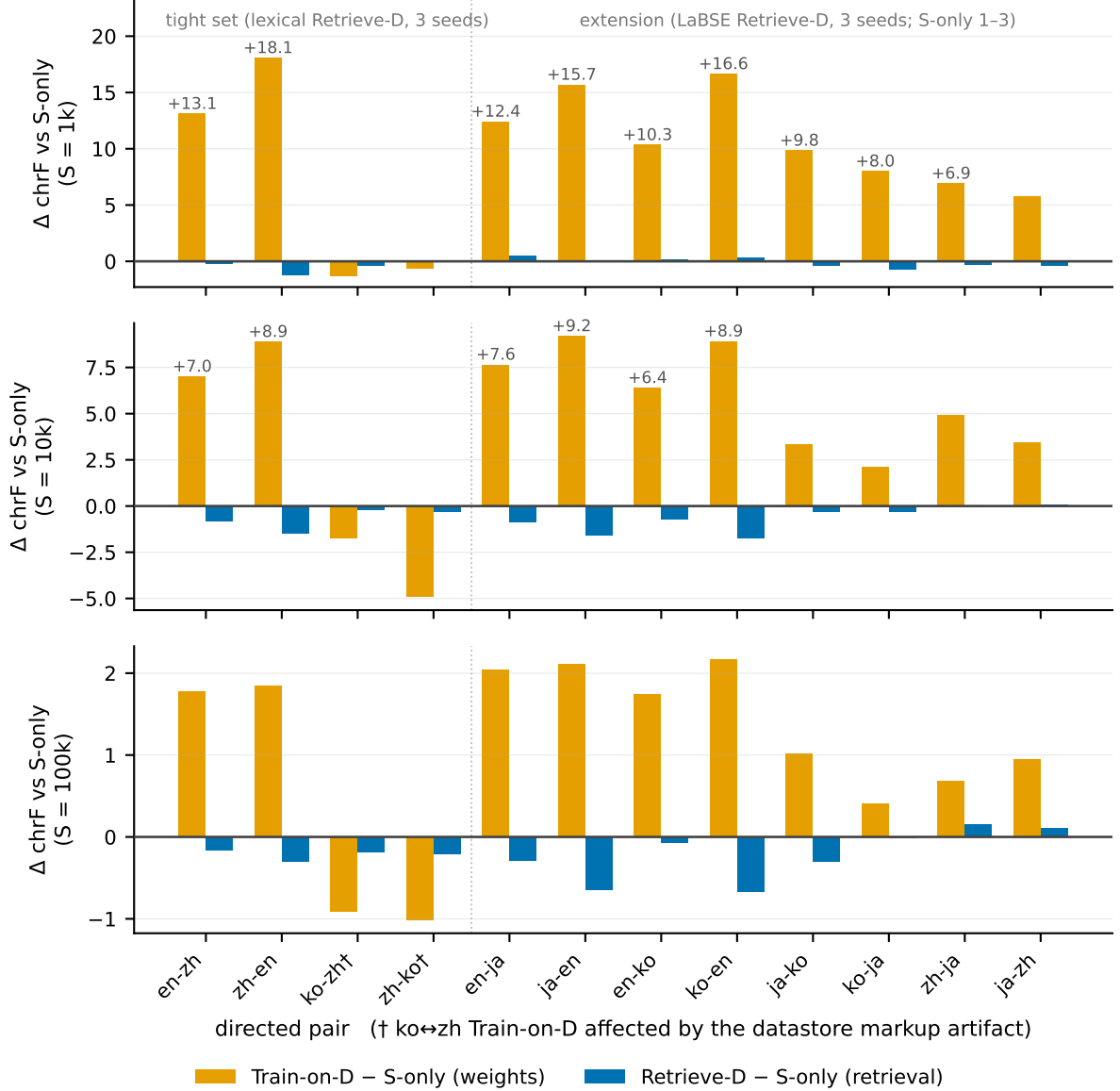


Figure 4: Datastore effects under parametric training and prompt-level retrieval across all 12 directed pairs. Each panel shows TRAIN-ON-D–S-ONLY and RETRIEVE-D–S-ONLY at one fine-tuning budget. The left side uses the lexical retriever on the four-pair core set; the right side uses the LaBSE variant from the eight-pair extension (Table 2). The figure combines the deltas in Tables 1 and 2.

tokens. The no-repeat constraint was selected in a pilot on ko→zh TRAIN-ON-D at the 100k budget after observing degenerate loops. It reduced looped outputs from 27/1,012 to 0 and outputs reaching the 256-token limit from 20 to 0; chrF increased from 13.72 to 14.47 and BLEU from 14.07 to 14.53. The setting was then fixed for all reported evaluations. Because the pilot used FLORES devtest, Section 5.2 additionally reports confirmatory results on FLORES dev.

Prompt construction and retrieval depth. The primary TF-IDF retriever uses no externally pretrained parameters and therefore satisfies the

corpus-access accounting. LaBSE is evaluated separately as an externally pretrained dense comparison. Each training instance is rendered as retrieved context, followed by the source sentence, a translation marker, and the target; loss is computed only on target tokens. Sentence-retrieval prompts contain the top $k = 3$ datastore examples. Concretely, each instance is rendered as

```
<|translate|> [SRC] → [TGT]
Examples:
1. [<src1>] → [<tgt1>]
2. [<src2>] → [<tgt2>]
3. [<src3>] → [<tgt3>]
<|src|> <source>
```

<|tgt|> <target>

where SRC and TGT are upper-cased language codes, the exemplars are numbered 1., 2., 3. and rendered as bracketed [source] → [target] lines with the arrow U+2192, and <|tgt|> is the translation marker; the order is exemplars, source, marker, target, with the target following the marker on the same line. RANDOM-CONTEXT replaces them with three count-matched random examples under the same template. Context construction is deterministic for a fixed seed, and random controls use per-sentence deterministic sub-seeds. Figure 1 displays only the highest-ranked example for readability, whereas all reported scores use the complete $k = 3$ prompt.

D Metrics and Statistical Analysis

Metrics. The headline metric is plain chrF (sacreBLEU CHRF, word_order=0), with BLEU, TER, COMET (Unbabel/wmt22-comet-da), and chrF++ (word_order=2) as secondary metrics, using language-aware tokenization for BLEU (sacreBLEU tokenize=zh for Chinese targets; character-level, tokenize=char, for Japanese and Korean). All metrics are computed from saved per-sentence predictions in a single scoring pass per metric across every system.

Metric consistency audit. An initial grid collation labeled chrF++ values as chrF for one subset of systems, whereas the attribution systems used plain chrF. Because the metrics differ by 2–4 points on English and Chinese targets, all systems were rescored from saved per-sentence predictions using a single implementation. Recomputed chrF++ reproduced the original values for all 486 evaluated cells, confirming that the discrepancy was a labeling error rather than a prediction mismatch. All values reported in the paper use the rescored metrics.

Metric robustness on the core set. Rescoring the four core conditions with BLEU, TER, and COMET preserves the qualitative ordering in Section 5.2. At the 100k budget, RETRIEVE-D–S-ONLY is at most +0.02 BLEU and +0.007 COMET for every pair, NO-CONTEXT is 0.19–0.28 COMET below RETRIEVE-D, and RANDOM-CONTEXT is within 0.016 COMET of RETRIEVE-D.

Significance. Primary contrasts are within-pair paired deltas on the same FLORES sources: the attribution deltas RETRIEVE-D–S-ONLY and TRAIN-ON-D–S-ONLY, the allocation delta

RETRIEVE-D–TRAIN-ON-D, and the dependence delta RETRIEVE-D–NO-CONTEXT. For each contrast we aggregate per-seed deltas across the 3 seeds with a Student- t 95% confidence interval (significant iff the interval excludes zero) and control the false discovery rate across the family of per-seed tests (Benjamini and Hochberg, 1995); the FDR-adjusted results agree in sign and significance with the Student- t intervals reported in the tables (no test changes status under correction). Per-run paired bootstrap intervals were also computed but not preserved, so only the Student- t and FDR results are reported. Cross-pair comparisons are descriptive only, since data sources and domains differ across pairs.

E Confirmatory Evaluation on FLORES-200 Dev

Motivation for length matching. RANDOM-CONTEXT matches the number of exemplars but not their rendered length. At the 100k budget, mean prompt length under RETRIEVE-D versus RANDOM-CONTEXT is 158 versus 249 tokens for en–zh and 183 versus 251 for zh–en. LM-RANDOM therefore matches the three exemplar lengths exactly.

Run provenance. The confirmatory runs are inference-only evaluations of preserved checkpoints using the decoding configuration in Appendix C. Each run records hashes for the checkpoint, tokenizer, datastore, and evaluation data. On audited zh–en 100k cells, the S-ONLY, RETRIEVE-D, RANDOM-CONTEXT, and NO-CONTEXT paths reproduced all 1,012 saved devtest hypotheses byte-for-byte. TRAIN-ON-D was reproduced within 0.06 chrF at corpus level; its saved hypotheses were generated with a different batch size and are therefore not byte-comparable.

Confirmatory split. FLORES-200 dev was obtained from the official NLLB release. As a data-integrity check, the corresponding devtest files reproduce the original evaluation files exactly after trailing-whitespace normalization. A repository audit found no FLORES-dev download, cache, configuration, code path, or notebook reference predating this analysis; we therefore treat dev as an unused confirmatory split.

LM-RANDOM construction. For each dev source, ordinary lexical retrieval produces three exemplars, each rendered into the prompt as a numbered [source] → [target] line; the rendered

pair	S-only	Train-D	Retr-D	Rand-ctx	LM-Rand	No-ctx	Δ Train	Δ Retr
zh-en	35.04	36.78	35.17	35.04	34.92	18.30	+1.75	+0.13
en-zh	15.97	17.82	15.83	15.64	15.69	4.96	+1.93	-0.14
ko-zh	14.21	13.04	14.02	13.95	—	4.64	-1.17	-0.19
zh-ko	14.79	14.10	14.90	14.74	—	3.78	-0.67	+0.11

Table 8: Confirmatory re-evaluation on FLORES-200 dev ($n = 997$, $S = 100k$, plain chrF; conditions as in Figure 2, LM-Rand = LM-RANDOM). S-ONLY, RETRIEVE-D, RANDOM-CONTEXT, NO-CONTEXT are 3-seed means; LM-RANDOM additionally averages five random draws per seed. TRAIN-ON-D is available for seeds 2–3 only (its checkpoint was inadvertently deleted before the re-evaluation; its saved devtest predictions survive and enter Table 1) and is reported descriptively. Δ Train is the mean paired delta over those two seeds, so it does not equal the difference between the displayed 3-seed S-ONLY mean and the 2-seed TRAIN-ON-D mean; both paired deltas have the same sign in every pair (e.g., en-zh +1.95/+1.91), but we do not treat the two-seed interval as a stable uncertainty estimate. All ko $\leftrightarrow$ zh retrieval-conditioned cells inherit the datastore markup artifact of Appendix G (the checkpoints were not retrained) and are not clean causal comparisons, though the observed ordering is stable across splits.

lines are tokenized with the checkpoint’s own BPE to give a length vector $\ell = (\ell_1, \ell_2, \ell_3)$. Replacements are drawn from the same per-pair datastore to minimize $\|\ell - \ell'\|_1$, preferring exact per-position matches, and excluding the true retrieved examples, any normalized self-match of the source, and duplicates within a prompt, the same exclusion rules as RANDOM-CONTEXT. With $\approx 167k$ candidate pairs per datastore, an exact length-vector match was found for 100% of the 29,910 prompts (median and maximum L_1 distance 0), and the realized mean prompt length is identical to RETRIEVE-D’s. Five draws per sentence and checkpoint seed use deterministic hash-derived seeds; draw-level variability is small (per-seed draw standard deviation ≤ 0.11 chrF) and is reported separately from seed variability. One caveat: the draw seed includes the checkpoint seed, so different fine-tuning seeds see different random exemplars, and the across-seed interval for LM-RANDOM–RETRIEVE-D includes residual context-sampling variation in addition to fine-tuning-seed variation.

NO-CONTEXT empty outputs. On zh $\rightarrow$ ko dev, 238–346 of 997 stripped-context outputs are empty strings (across the three seeds); the other pairs produce none. Empty hypotheses are preserved in the saved predictions and included in all metric computations. Removing context creates an out-of-distribution prompt mismatch, so the NO-CONTEXT score collapse partly reflects generation failure; we read NO-CONTEXT as demonstrating dependence on the retrieval-conditioned interface, not productive use of retrieved information (consistent with the list-marker degeneration on zh $\rightarrow$ en documented in Appendix H).

F Dictionary Retrieval Analysis

Design. DICT-RETRIEVE tokenizes the source sentence (language-aware: kiwipiepy for Korean, jieba for Chinese, fugashi for Japanese), looks up exact lowercased dictionary matches, keeps only rare words, and caps at a fixed number of entries. “Rare” is a rank-based bottom-quartile gate: within the active fine-tuning budget split, unique source words are ranked by frequency and the rarest 25% are eligible. This same rare-word set drives both the retrieval gate and the rare-word-accuracy metric used in evaluation, so the two are consistent by construction.

Coverage by fine-tuning budget. Eligibility is defined relative to each fine-tuning subset and also requires dictionary coverage. Under training-time augmentation of the fine-tuning split, the fraction of sentences receiving at least one entry falls from approximately 30–60% at 1k to below 3% at 100k across the ten directions with a usable lexicon; the evaluation-time densities reported below for English-source directions (2.4 entries per sentence at 1k) measure the same gate on FLORES. The lowest-frequency quartile of the larger vocabularies contains many hapax forms, proper nouns, and typographical variants not covered by the fixed lexicons. Coverage is already minimal at 1k for ja $\rightarrow$ en (0.022 entries per sentence) and ja $\rightarrow$ zh (0.007); the latter is excluded from the dictionary family, together with zh $\rightarrow$ ja, because the family is dropped for the zh $\leftrightarrow$ ja pair as a whole (Appendix A).

Evaluation-time tokenizer mismatch. Training-time augmentation used the intended language-specific tokenizers. During evaluation, however, the frequency map was reconstructed

with an English tokenizer. Its keys do not match Korean, Chinese, or Japanese lemmas, which disables the rarity gate for the seven CJK-source directions. These directions therefore receive approximately five entries per sentence on nearly all examples at every budget. English-source directions follow the intended density schedule (2.4, 0.6, and 0.08 entries per sentence at 1k, 10k, and 100k). ORACLE-DICT is gated by the reference and remains approximately budget-independent; RANDOM-DICT matches the realized DICT-RETRIEVE count in each cell, so their direct contrast remains paired.

Interpretable subset. The en→CJK directions are the only dictionary conditions for which the training and evaluation tokenizers agree. At the 1k budget, where injection is non-trivial, all three pairs satisfy $\text{ORACLE-DICT} \geq \text{DICT-RETRIEVE} \geq \text{RANDOM-DICT}$ (e.g., 4.20, 4.12, and 3.58 chrF for en→zh). At 10k and 100k, the intended gate injects too few entries for a reliable retrieval-quality comparison. The CJK-source directions instead contain a train–evaluation context-density mismatch of up to approximately $50\times$ at 100k. This mismatch is associated with template copying in ja→ko at 100k (53.7% of outputs) and with inversions between ORACLE-DICT and DICT-RETRIEVE in some cells. We therefore exclude all affected CJK-source dictionary conditions from retrieval-quality and mechanism claims in the main text.

G ko↔zh Datastore Artifact Analysis

ko↔zh is the only pair family for which TRAIN-ON-D is below S-ONLY at every budget. We identified a leading-> markup artifact in approximately 50% of ko↔zh datastore target sentences; the artifact is absent from the fine-tuning subsets and FLORES. It enters TRAIN-ON-D as a training target and enters RETRIEVE-D and RANDOM-CONTEXT only as conditioning context. To isolate its effect on parametric training, we retrained TRAIN-ON-D after removing the marker from D . S-ONLY does not access D , and the retrieval-conditioned checkpoints were not retrained. Table 9 reports the resulting TRAIN-ON-D–S-ONLY deltas.

At zh→ko 10k, cleaning increases the mean delta by 2.2 chrF: two seeds recover to within 1.6 chrF of S-ONLY, while one remains in the low-performing regime. In the other five cells, cleaning changes the delta by at most 0.3 chrF. Residual deficits of approximately 0.4–1.7 chrF remain in

pair	budget	dirty	cleaned	moved
ko-zh	1k	−1.3	−1.6	≈0
ko-zh	10k	−1.7	−1.7	≈0
ko-zh	100k	−0.9	−1.0	≈0
zh-ko	1k	−0.6	−0.4	≈0
zh-ko	100k	−1.0	−1.1	≈0
zh-ko	10k	−4.9	−2.7	+2.2

Table 9: $\Delta = \text{TRAIN-ON-D} - \text{S-ONLY}$ before and after removing the leading-> marker from the ko↔zh datastore (plain chrF). The zh→ko 10k row is placed last for emphasis. The cleaned 10k cells are 3-seed means; the cleaned 1k and 100k cells are single-seed. At zh→ko 10k, all three original runs have chrF 6.2–6.7; two of three cleaned runs recover to the range of neighboring cells (their mean Δ is −1.6), while the third does not, giving the 3-seed mean of −2.7. Deltas and movements are computed from unrounded means.

the other five cells; zh→ko at 10k remains 2.7 chrF below S-ONLY because one cleaned seed does not recover. Because the retrieval-conditioned checkpoints were not retrained, their ko↔zh results remain artifact-bearing and are interpreted as diagnostic rather than fully controlled comparisons.

H Qualitative Example Selection

Reported values. The score strip in Figure 1 reports three-seed mean corpus chrF for zh→en at $S = 100k$: 34.8 for S-ONLY, 34.4 for lexical RETRIEVE-D, 34.4 for RANDOM-CONTEXT, and 18.1 for NO-CONTEXT; TRAIN-ON-D is omitted from the figure and reported in Table 1. The displayed sentence outputs are from seed 1. All figure values use the lexical retriever; the ko↔zh allocation contrasts are reported in Table 5.

Selection procedure. For each of the 1,012 zh→en devtest items, we compute sentence-level plain chrF averaged over seeds for S-ONLY, lexical RETRIEVE-D, RANDOM-CONTEXT, and NO-CONTEXT. An item is eligible when (i) $|\text{RETRIEVE-D} - \text{RANDOM-CONTEXT}| \leq 10$, (ii) $\min(\text{RETRIEVE-D}, \text{RANDOM-CONTEXT}) - \text{NO-CONTEXT} \geq 20$, (iii) $|\text{RETRIEVE-D} - \text{S-ONLY}| \leq 10$, and (iv) the reference contains 8–30 words. These criteria yield 200 items with a median context-removal gap of 26.7 chrF. Figure 1 uses the item nearest this median (index 677; gap 26.9). The figure shows only the highest-ranked of the three retrieved examples.

No-context output pattern. Among the 1,012 zh→en NO-CONTEXT outputs, 232 (23%) contain repeated list markers such as “(iii) (ii) (iv)...”. Most remaining outputs are fluent but unrelated

to the source, as in the displayed example.

I Validation and Reproducibility Checks

Control checks. For the dictionary analysis, we verify at the 1k budget that oracle retrieval is at least as accurate as non-oracle dictionary retrieval on the rare-word metric and that random dictionary context does not reproduce the same gain. These checks are interpreted only for the English-source directions because the CJK-source conditions contain the tokenizer mismatch described in Appendix F.

Budget validation. Before training, $P + S + D + \text{dict}$ is computed with the directed model’s tokenizer over both sides of the deduplicated records. The datastore is then capped at $100\text{M} - P - S - \text{dict}$. This validation identified initial pair-count-based datastores for $\text{en} \leftrightarrow \text{zh}$ and $\text{en} \leftrightarrow \text{ja}$ that exceeded the limit by 0.7–1.8M tokens; Appendix A reports the corrected token caps.

Oracle isolation. ORACLE-DICT uses the reference translation and is restricted to evaluation. Module-level, constructor-level, and AST checks prevent the oracle component from being imported or instantiated by the training pipeline.

Determinism. Given a fixed seed, the augementer produces identical context for a given source sentence across runs. RANDOM-CONTEXT and RANDOM-DICT each use a per-sentence deterministic sub-seed, so their controls are themselves reproducible.

Saved predictions. All metrics are computed from stored per-sentence hypotheses. This permits consistent rescoring across metrics and enabled the metric-label and dictionary-gate audits described in Appendices D and F.