DEMYSTIFYING CACHE-TO-CACHE COMMUNICATION: WHEN LATENT TRANSFER REDUCES TO OUTPUT CALIBRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work proposes *cache-to-cache* (C2C) communication for heterogeneous LLM agents, directly transferring KV caches to convey question-specific semantics more efficiently than text-based communication. In this paper, we revisit this mechanism and uncover a different explanation: C2C gains arise primarily from a *question-independent distributional shift* that corrects a degenerate receiver output distribution, rather than from question-specific semantic transfer. Using the authors' released weights and evaluation protocol, we first reproduce the reported gains, then conduct four controlled experiments. First, in a setting where only the sender holds the answer, nothing crosses between the released model pair. Second, replacing the sender KV cache with zeros preserves nearly the full improvement, showing that sender-specific semantic content is unnecessary. Third, decomposing the induced logit shift shows that a single question-independent vector reproduces the gain, while the question-dependent residual is ineffective or harmful. Fourth, varying receiver scale shows that gains appear only when the receiver output distribution is degenerate and disappear after directly centring it. The effect has a boundary: fusers trained on task data do carry the sender's answer when the sender is much stronger or the receiver has no bias to correct. Together, these results challenge the interpretation of C2C as semantic communication and instead attribute its benchmark gains to a content-independent correction of the receiver distribution. Our findings highlight the need to distinguish semantic information transfer from distributional calibration when evaluating latent communication between LLMs. Code, per-item outputs and the data of the constructed tasks are available at `https://anonymous.4open.science/r/gains-without-transfer`.

1 INTRODUCTION

Communication is central to multi-agent systems built on large language models (LLMs), where agents exchange information to coordinate their reasoning (Du et al., 2024; Wu et al., 2024; Guo et al., 2024). In a typical communication process, a *sender* conveys its knowledge to a *receiver*, which incorporates the received information to solve a task. Text provides a common interface between heterogeneous models, but requires the sender to decode its internal representations into tokens and the receiver to process these tokens through an additional prefilling step. This process can discard information encoded in the sender's latent states and incur substantial computational overhead (Pham et al., 2024; Ramesh & Li, 2025; Zou et al., 2026).

Cache-to-cache (C2C) communication (Fu et al., 2026) offers a promising alternative by directly transferring the sender's key-value (KV) cache to the receiver: a learned *fuser* maps the sender's cache into the receiver's cache space and adds it to the receiver's own cache. Later methods, LCF (Rossi et al., 2026) and XKV (Liu et al., 2026a), keep this fuser design, and all three report their gains with receivers of about 1B parameters. Existing work reports that C2C can improve both communication efficiency and task performance (Fu et al., 2026; Rossi et al., 2026), suggesting a new foundation for multi-agent collaboration through direct semantic exchange.

This potential makes it essential to understand what information C2C actually transfers. Prior work attributes the observed performance gains to the transfer of question-specific semantics from

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

the sender to the receiver (Fu et al., 2026; Rossi et al., 2026; Liu et al., 2026a). However, the receiver's accuracy improvement alone does not establish that the sender's knowledge has crossed the communication channel. The learned cache-fusion mechanism could instead alter the receiver's output distribution in a way that improves benchmark performance without conveying sender-specific content. For example, it could correct the receiver's persistent bias toward particular answer options (Zhao et al., 2021; Zheng et al., 2024; Pezeshkpour & Hruschka, 2024) even when the transferred representation contains no information about the current question. Semantic transfer and output calibration can therefore produce similar accuracy gains while representing fundamentally different communication mechanisms. This raises a central question:

> *Do the reported gains of C2C truly arise from transferring question-specific information between agents? If not, what mechanism actually drives these gains?*

Two recent audits have begun to examine latent communication, including C2C, from a causal perspective (Zhang & Emu, 2026; Cheng et al., 2026), but they leave open what produces C2C's benchmark gains. In this paper, we revisit C2C through a controlled empirical study. Using the original authors' released weights and evaluation protocol, we first reproduce the reported gains. We then ask one question: does the performance gain survive when the sender's content is removed? If the receiver benefits from what the sender knows, the gain should disappear; if the fuser only corrects the receiver's own bias, it should remain.

Four controlled experiments reveal a counter-intuitive explanation for the reproduced gains. ❶, the *Answer-Exclusive Transfer Test* places the answer exclusively with the sender and finds that the receiver cannot recover it through the released C2C model pair, providing no evidence of information transfer. ❷, the *Content-Free Cache Test* replaces the sender's KV cache with zeros yet preserves nearly the full accuracy improvement, showing that sender-specific semantic content is unnecessary for the observed gains. ❸, the *Logit-Shift Decomposition* starts from what zeroing leaves open: the fuser still reads the receiver's own cache and could adjust each question differently. We split its change to the option scores into a global average (μ) and question-specific deviations. μ alone recovers nearly all of the gain, while the question-specific deviations lower accuracy. ❹, the *Cross-Scale Checkpoint Audit* evaluates the released fusers across model scales (0.6B, 1.7B, and 8B). We find that gains occur *exclusively* on the 0.6B receiver, the only one of the three with selection bias. On 1.7B and 8B receivers that already exhibit balanced distributions, the released fusers provide zero benefit and instead degrade accuracy by up to 8.8 points. Together, these experiments reveal that the evaluated C2C system improves performance primarily by calibrating the receiver's output distribution rather than by transferring question-specific information.

Our contributions are summarized as follows:

- **Mechanistic Evaluation.** We distinguish semantic information transfer from output calibration in C2C, examining whether performance gains depend on what the sender supplies.
- **Counter-intuitive Empirical Findings.** After reproducing the reported gains, four controlled experiments show that a question-independent distributional correction primarily explains the improvement in the released system.
- **A Boundary for the Calibration Explanation.** We identify conditions under which trained fusers do carry content, clarifying when C2C supports semantic communication and motivating controls for evaluating latent transfer. Section 5 shows that the other methods of this kind use the same kind of receiver and report no such control, and gives two cheap checks that tell transfer from calibration.

2 Background and related work

As shown in Figure 1, sharing a KV cache between language models is done for two different purposes. The first line treats the cache as a message: one model's cache is projected into another model's cache space so that the second model can answer better than it would alone (Fu et al., 2026; Rossi et al., 2026; Liu et al., 2026a). The second line treats the cache as a computation to be saved: a prefix cache produced once is reused, within or across models, so that the prefix need not be

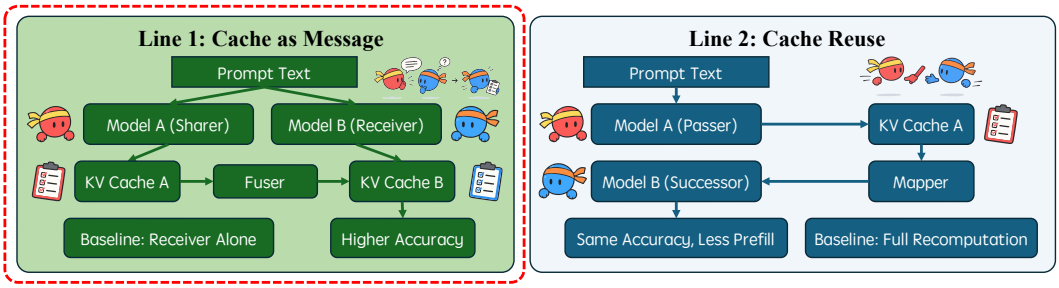


Figure 1: Two uses of KV-cache sharing. In Line 1 the sender's cache is added to the receiver's, and success means higher accuracy than the receiver alone. In Line 2 a precomputed cache replaces the receiver's own prefill, and success means the same accuracy as full recomputation at lower cost.

recomputed (Heo et al., 2026; Liu et al., 2024; Yao et al., 2025; Woo et al., 2026). We focus our audit on the first line because it holds greater promise: it can remove the sender's decoding step altogether.

Cache-to-Cache (C2C) (Fu et al., 2026) is the representative of the first line: at every layer, a learned fuser adds the sender's cache to the receiver's (§3.1). The C2C paper reports that a fused receiver scores 8.5–10.5 points above the receiver alone and 3–5 points above a text-to-text baseline, and reads these gains as semantic information arriving from the sender. Latent Cache Flow (LCF) (Rossi et al., 2026) keeps this design with a shared 128-dimensional bottleneck and re-evaluates the released C2C checkpoint under a logit-scoring protocol; XKV (Liu et al., 2026a) adds a second cache pathway. All three report their main results with a receiver of about 1B parameters or less.

The second line reuses a prefix cache to save computation rather than to add information. Heo et al. (2026) is representative: a closed-form linear map carries the prefix cache of one model to another model of the same family, so that the second model skips prefill, and the method is judged by how little accuracy it loses against recomputing the prefix from scratch. Related systems reuse caches across contexts or across serving instances under the same criterion (Liu et al., 2024; Yao et al., 2025; Woo et al., 2026; Yang et al., 2025; Lee et al., 2026; Shi & Lou, 2026). A reused cache is measured against recomputation and can never be credited with accuracy the model did not already have.

The two lines make different claims. Cache reuse claims to lose no accuracy relative to recomputation; cache communication claims to add accuracy beyond the receiver alone, and that claim holds only if the added accuracy depends on what the sender sent. Accuracy can also rise without any sender: language models favour particular answer options (Zhao et al., 2021; Zheng et al., 2024; Reif & Schwartz, 2024), often the first one (Dominguez-Olmedo et al., 2024; Li & Gao, 2025), and removing that preference alone, as batch calibration does (Zhou et al., 2024), raises accuracy without adding any information. This paper audits the communication line as C2C, LCF and XKV publish it: position-wise, per-layer fusers from a sender's cache into a receiver's, evaluated by option accuracy on multiple-choice benchmarks with a receiver of about 1B parameters or less. Cache reuse, same-model relays (Zou et al., 2026; Ye et al., 2025; Shi et al., 2026), dense latent adapters (Chen et al., 2026), cross-attention adapters (Dery et al., 2026) and free-form generation are outside this scope.

3 Evaluation Setup

In this paper, we present an empirical evaluation that aims to address the following research questions.

1. **RQ1**: Does C2C indeed transmit information between sender and receiver?
2. **RQ2**: Does the C2C performance gain depend on the content sent by the sender?
3. **RQ3**: Where do the performance gains from C2C come from?
4. **RQ4**: How does agent model size affect the performance gains from C2C?

3.1 Experimental subjects

C2C algorithm under study. In this work, we focus on C2C (Fu et al., 2026) because it pioneered cache-to-cache communication and established the fuser design adopted by later methods such as

LCF (Rossi et al., 2026) and XKV (Liu et al., 2026a). It is also the only one of the three with released weights, which lets us audit the original system rather than a reproduction; for LCF we test our own reimplementation (Appendix H), and XKV has released no code. C2C lets a receiver read a sender's KV cache instead of a text message from it. Both models prefill the same prompt, and C2C attaches a fuser to every layer of the receiver. The fuser at receiver layer i concatenates the receiver's cache R_i with the cache S_i of the sender layer paired with it, passes the result through a projection and a fusion layer f_i, reweights it per head with an input-dependent weight α_i, and adds it to R_i through a learned gate g_i:

$$R'_i \;=\; R_i + g_i \cdot \big(\alpha_i \odot f_i([R_i \,\|\, S_i])\big). \tag{1}$$

Layers are paired from the top, the last sender layer with the last receiver layer, and tokens are aligned by decoding each receiver token to its string and re-encoding it with the sender's tokenizer. The gate is a Gumbel-sigmoid whose temperature is annealed during training, so that at inference each layer's injection is either on or off. Both language models stay frozen and only the fusers are trained, 478M parameters for the 0.6B receiver, with next-token loss on the receiver's responses over 500k OpenHermes-2.5 samples (Teknium, 2023). The receiver then answers from the fused prompt cache R', and the tokens it generates are cached as usual. Because f_i reads R_i as well as S_i, replacing the sender's cache with zeros or noise does not switch the fuser off but leaves a learned function of the receiver's own cache in place, and this is the property the control arms of RQ2 rely on. Cheng et al. (2026) instead zero the fuser's output.

Datasets. We use two constructed tasks for RQ1 and four public benchmarks for RQ2–RQ4. RQ1 needs tasks that the receiver cannot solve alone: on a public benchmark the receiver already answers many items correctly, so a gain does not show that anything was transferred. In each item of our tasks, the sender reads eight records, while the receiver reads the same prompt with every record value masked. In the *4-digit code repetition* task, a record reads *"The access code for TERMINAL_7 is 4821"* and the receiver must output the requested code, so guessing succeeds with probability 10^{-4}. In the *colour lookup* task, each record assigns an object to a colour, and the receiver must choose which of four listed objects has a given colour, so only a single option letter has to cross and guessing scores 25%. For RQ2–RQ4 we use the four benchmarks that the audited methods report: OpenBookQA with 500 items (Mihaylov et al., 2018), ARC-Challenge with 1172 (Clark et al., 2018), MMLU-Redux with 5632 (Gema et al., 2024) and MMLU-Pro with 12032 (Wang et al., 2024). The first three form the main table of C2C, and LCF (Rossi et al., 2026) added the fourth. We reproduce the published numbers on these sets (Table 6), and the fusers we retrain for RQ4 are evaluated on OpenBookQA and MMLU-Redux. All four benchmarks spread correct answers nearly evenly over the letters, so a receiver that favours one letter loses accuracy that bias correction can recover.

LLM agents. Following C2C (Fu et al., 2026), we use the three sender–receiver pairs for which the authors released fusers: Qwen3-0.6B, 1.7B and 8B receivers with Qwen2.5-0.5B, 1.5B and 7B-Instruct senders. Other models are named where they are used and listed in Appendix B. Unless stated otherwise, one scorer produces every number in this paper, the authors' released evaluator, which appends *"The correct answer is"* to the chat template and takes the option whose first token has the highest logit.

3.2 Experiment process and metrics

RQ1 Process. RQ1 asks whether the C2C channel can carry sender's information. Using the authors' released code, architecture and training recipe, we train the C2C fuser between the released pair on the two constructed tasks (§3.1). In both tasks, the sender reads a set of records that contains the answer. The receiver reads the same records with the answer masked. It can therefore answer only with what the sender passes through the fuser. We train the fuser on these tasks rather than reuse the released weights, so a failure cannot be blamed on unfamiliar data. To check that the task is learnable, we train the same implementation between two Qwen3 models. If the receiver scores clearly above chance, the channel carries information. The next token the sender would itself output is its *verdict*; the second to fourth digits of a code can cross only as *content*. If it stays at chance, we classify the pair as *unable to relay content*.

RQ2 Process. RQ2 asks whether the benchmark gain depends on what the sender sends. We keep the fuser released by the C2C authors fixed and replace the sender's cache with zeros (ZERO), the cache with shuffled token positions (SHUFFLE), or Gaussian noise. Zeros and noise carry no information

about the question; shuffling keeps its tokens but not their order. If the gain disappears, we classify the fuser as *content-dependent*. If the gain stays, we classify it as *content-independent*.

RQ3 Process. RQ3 asks where the gain comes from. Without the sender, the fuser still reads the receiver's own cache, so it could still treat each question differently. We look at the logits of the option letters (A, B, C, ...) at the position where the receiver gives its answer. For each question x, let $\Delta(x)$ be how much the fuser changes these logits. We split $\Delta(x)$ into two parts (Eq. 2):

$$\Delta(x) = \ell_{\text{fused}}(x) - \ell_{\text{recv}}(x) = \mu + E(x), \qquad \mu = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \Delta(x). \tag{2}$$

The first part, μ, is the average of $\Delta(x)$ over all questions, so it is the same for every question. The second part, $E(x) = \Delta(x) - \mu$, is what remains for each question. We add each part to the receiver alone. We also subtract the receiver's own mean option logits without any fuser (*centring*). On benchmarks whose items differ in the number of options, both are indexed by option letter, and options an item lacks never enter its prediction. If $E(x)$ carries the gain, we classify it as *question-specific*. If μ or centring alone reproduces it, we classify it as a *uniform shift*.

RQ4 Process. RQ4 asks how model size affects the gain. Small language models often show *selection bias* (Zheng et al., 2024): *they favour one option letter regardless of the question*. Such a receiver has a low-entropy answer distribution. Before training, we measure two quantities: the entropy of the receiver's answer distribution, and how much more accurate the sender is than the receiver. We evaluate the released 1.7B and 8B fusers and train five fusers with receivers from 0.6B to 4B and senders from 0.5B to 7B. If the zero cache keeps the gain, we classify the fuser as *correcting the receiver*. If the fused receiver copies the sender's answers, we classify it as *following the sender*.

Implementation details. Fusers we train use the released C2C code, architecture and layer mapping, with both backbones frozen: AdamW at 10^{-4}, 1,000 steps of 32 sequences, and Gumbel-sigmoid gates annealed from 1.0 to 0.001. The fusers of RQ4 instead run one epoch over the first 40,000 items of MMLU auxiliary-train, the split used by the C2C paper's receiver-scale sweep. Intervals come from a paired bootstrap over per-item correctness with 10,000 resamples, and equivalence is tested by TOST at $\delta = \pm 2$ points with Holm correction over 20 comparisons.

4 Results

4.1 RQ1: Does C2C indeed transmit information between sender and receiver?

The released pair fails to transmit private content. Table 1 shows that the fuser trained between Qwen2.5-0.5B-Instruct and Qwen3-0.6B never beats chance, while the same implementation, task and recipe transfer almost every code between two Qwen3 models. On the 4-digit code repetition task, its training loss remains at $\ln 10$ and the receiver achieves 0.00% exact match. Replacing the sender's cache with zeros, sequence shuffles, or mismatched items yields identical chance performance. Further variations—including initialising from the released fuser, tripling the training budget, and fixing the gates open—fail to lift accuracy above 0.05% (Appendix G). The same failure holds on the colour lookup task (25.35% vs. 25% chance), although there the 0.5B sender alone scores only 27.8%. This extends the 8B null of Cheng et al. (2026) to the main-table pair, with fusers trained on the task.

The bottleneck is representation mismatch, not the pipeline. Under the identical training recipe, the channel succeeds when representations align: the receiver achieves 95.45% exact match when communicating with another Qwen3-0.6B, and 97.95% with Qwen3-1.7B. Here, zeroed or shuffled caches immediately collapse accuracy back to chance, confirming that the code crosses the channel. The architecture and code can carry content; what fails is crossing the representation gap between Qwen2.5 and Qwen3. Cache width is not the cause: the 7B sender below has four times the width.

Scaling the sender transfers only its final verdict, not the underlying content. Replacing the sender with the much stronger Qwen2.5-7B-Instruct does not bridge this gap. Although the receiver now predicts the *first digit* with 98.9% accuracy, accuracy on the remaining three digits collapses to chance ($\sim$10%), leaving exact match stuck at 0.00%. This first digit is the next token the 7B sender would output itself: feeding a mismatched cache causes the receiver to output that wrong item's first digit at the same rate (98.9%). The same phenomenon governs the colour lookup task: the receiver matches the 7B sender's high standalone accuracy (97.85% vs. 97.30%).

Table 1: Can the fuser pass on information that only the sender has? Constructed tasks of §3.1, n=2,000 each. *Exact*: all four digits correct (chance 0.01%). *Entropy* of the receiver's answers; uniform guessing gives 2.30 (codes) and 1.39 (colours). *Controls*: another item's cache / zeros / a shuffled cache. More variants in Appendix G; —: not run.

		Task: 4-digit code repetition		Task: colour lookup (4 options)	
Sender → receiver	Condition	Exact (%)	Entropy	Acc. (%)	Entropy
Reference	Receiver, records masked (floor)	0.00	1.19	25.70	1.127
	Receiver, records shown	85.60	0.26	64.90	0.940
	Sender alone, 0.5B / 7B	78.80 / 99.75	0.34 / 0.00	27.80 / 97.30	1.264 / 0.024
Released pair	sender's cache	0.00	2.30	25.35	1.385
0.5B → 0.6B	controls	0.00	2.30	25.1–25.6	1.385
Same family	sender's cache	**97.95**	0.01	—	—
1.7B → 0.6B	controls	0.00 / 0.05 / 0.00	0.61 / 1.80 / 1.04	—	—
Stronger sender	sender's cache (1st digit: 98.9%)	0.00	1.73	**97.85**	0.055
7B → 0.6B	controls	0.00 / 0.00 / 0.00	1.73 / 2.13 / 1.99	23.7 / 26.5 / 25.8	0.16 / 1.07 / 0.58
Same model	sender's cache	**95.45**	0.05	**90.25**	0.210
0.6B → 0.6B	controls	0.00 / 0.00 / 0.05	0.68 / 1.94 / 0.97	23.2 / 24.9 / 25.7	0.35 / 0.80 / 0.69

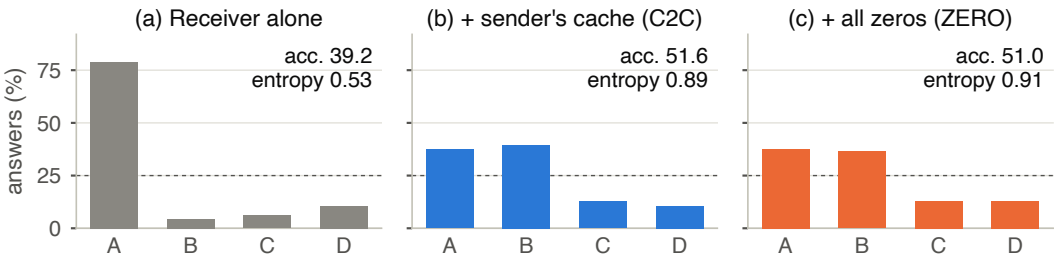


Figure 2: Answers of the released Qwen3-0.6B receiver on OpenBookQA. Alone, it picks "A" on 79% of items (a). The fuser spreads its answers with the sender's cache (b), and just as much with all zeros (c). Dashed line: uniform (25%).

4.2 RQ2: Does the C2C performance gain depend on the content sent by the sender?

Zeroing the sender's cache preserves nearly the entire gain. We keep every weight of the released fuser fixed and replace the sender's cache with three controls (§3.2): all zeros (ZERO), the same cache with shuffled token positions (SHUFFLE), and moment-matched Gaussian noise (GAUSS). ZERO retains 95–111% of the original C2C gain, and SHUFFLE retains 98–110% (Table 2)—on three benchmarks, both outperform the real sender cache, and ZERO moves the receiver's answers away from "A" as much as the real cache does (Figure 2). Only unstructured noise (GAUSS) lowers it (63–80% of the gain retained): the fuser needs a benign input, not the sender's content.

Real and zeroed caches induce nearly identical predictions item by item. At the individual item level, the real and ZERO caches yield the exact same option predictions on 92.1–95.4% of questions; whenever the real cache flips the receiver's original choice, ZERO makes the identical change in 85.9–92.9% of cases (Appendix C). This holds across setups: across 20 paired comparisons spanning three receiver scales, the real cache never statistically outperforms ZERO (Table 13). The C2C paper contains its own version of this control: with the receiver itself as sender ("Identical"), OpenBookQA reaches 50.60 against 52.60 with the real sender (Fu et al., 2026), so 85% of the gain over the receiver alone arrives without any new information.

4.3 RQ3: Where do the performance gains from C2C come from?

The fuser applies a uniform, question-independent logit shift. Since the gain does not come from the sender, it must come from how the fuser changes the receiver's logits, captured by the displacement $\Delta(x) = \ell_{\text{fused}}(x) - \ell_{\text{recv}}(x)$. If the fuser were performing adaptive reasoning, this shift would vary from question to question. Instead, geometric analysis reveals that displacements across

Table 2: Does the gain need the sender's content? Accuracy (%) of the released Qwen3-0.6B receiver (sender Qwen2.5-0.5B-Instruct). Bold: above the real cache. Last row: correct if either model is correct, the ceiling for passing on the sender's answers when neither model is calibrated (Appendix D).

What the fuser receives	OpenBookQA	ARC-C	MMLU-Redux	MMLU-Pro
No fuser (receiver alone)	39.20	39.33	34.91	15.58
Sender's real cache (C2C)	51.60	54.01	42.38	19.10
Shuffled cache (SHUFFLE, 3 seeds)	51.40 ± 0.53	**55.49** ± 0.38	**43.08** ± 0.37	**19.36** ± 0.05
All zeros (ZERO)	51.00	**55.12**	**43.18**	**19.41**
Gaussian noise (GAUSS)	47.00	51.11	40.23	18.33
Oracle: either model correct	56.00	54.10	48.97	20.89

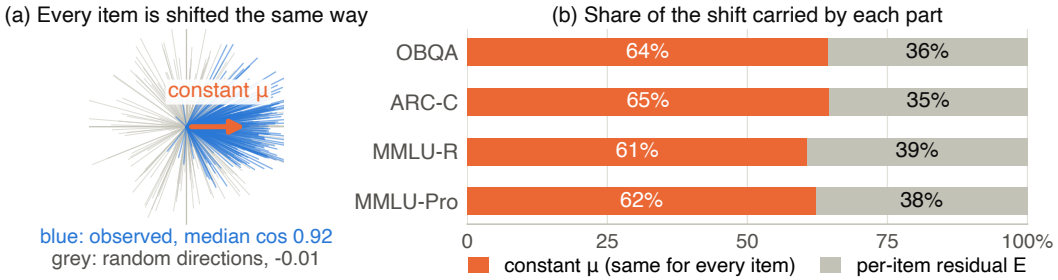


Figure 3: The intervention does the same thing to every item. (a) Logit displacements Δ_i of 200 OpenBookQA items (blue), projected onto the plane of μ (orange arrow) and the leading residual direction; grey: the same with random directions. (b) Share of displacement energy in μ and E.

different questions point in nearly the same direction (Figure 3a, median cosine 0.85–0.92), with a single dataset-mean vector μ accounting for 61–65% of the total displacement energy (Figure 3b).

We therefore decompose the displacement into this uniform shift μ and a question-specific residual: $\Delta(x) = \mu + E(x)$. Table 3 confirms what this geometry implies: adding the static vector μ alone to the receiver's logits recovers 93–199% of the full C2C gain across all four benchmarks. In contrast, adding the question-specific residual $E(x)$ alone—the only term capable of conveying problem-specific reasoning—consistently degrades accuracy to 2.9–8.7 points below the receiver alone. Adding $E(x)$ back to μ, which gives C2C, changes accuracy by -3.5 to $+1.1$ points.

The root cause: the receiver favours option "A". A constant shift can raise accuracy only if the receiver has a constant bias. The 0.6B receiver picks "A" on 79.0% of OpenBookQA items (Figure 2a; Appendix B): it is right on 92.8% of the items whose answer is "A" but on only 14–28% of the others, while gold answers are spread roughly evenly over the letters. The bias is not an artefact of logit scoring: under free-form decoding the receiver answers "A" on 78.6% of items. The shift μ is this bias reversed: on OpenBookQA it lowers option "A" by 3.9 logits, against the receiver's average excess of 4.3.

Training-free calibration rivals C2C, while pre-centring breaks it. If the fuser merely corrects an output bias, a 478M-parameter module is not needed. We test this with centring (§3.2): we compute the receiver's mean option logits on held-out items and subtract them at inference time—using no sender, no fuser, and zero training. It rivals C2C: it beats C2C on MMLU-Redux (44.35 vs. 42.38) and MMLU-Pro (23.97 vs. 19.10), while coming within 2.6 points on the other two (Table 3).

Conversely, applying C2C to an already-centred receiver causes accuracy to collapse by 6–11 points (Table 3, last column), driving predictions right back onto a single letter (54–76% share). Because the fuser learned an offset tailored to the receiver's bias, applying it to a calibrated model overshoots.

4.4 RQ4: How does agent model size affect the performance gains from C2C?

Without a bias to correct, the released fusers stop helping. If C2C's gain is spent on correcting the receiver's selection bias, it should disappear on receivers that have no such bias. The released

Table 3: Where the gain comes from, at receiver Qwen3-0.6B. Each part of the fuser's logit change (Eq. 2) is added to the receiver alone; *Centring* uses no sender and no fuser. Parentheses: change from the receiver alone (last column: from the centred receiver). Cross-fitted, 20 splits, s.d. ≤ 0.8. †Fitted *with the labels*: an upper bound for any bias correction.

			C2C shift, split in two		No sender, no fuser		
Benchmark	Receiver alone	C2C	Shared μ only	Per-item E only	Centring	Best bias†	Centred, then C2C
OpenBookQA	39.20	51.60(+12.40)	51.38(+12.18)	30.47(−8.73)	50.13(+10.93)	53.40	39.13(−11.00)
ARC-C	39.33	54.01(+14.68)	52.93(+13.60)	31.82(−7.51)	51.49(+12.16)	54.85	45.03(−6.46)
MMLU-Redux	34.91	42.38(+7.47)	43.62(+8.71)	28.10(−6.81)	44.35(+9.44)	45.65	36.13(−8.22)
MMLU-Pro	15.58	19.10(+3.52)	22.57(+6.99)	12.70(−2.88)	23.97(+8.39)	24.28	17.93(−6.04)

Table 4: Released fusers at three receiver sizes: change in accuracy (points) from the receiver alone with the sender's cache (*real*) or zeros (*zeros*). Senders: Qwen2.5-0.5B / 1.5B / 7B-Instruct. Entropy on OpenBookQA / ARC-C; 1 means answers spread evenly. Only the biased 0.6B receiver gains.

Receiver	Entropy	OpenBookQA		ARC-C		MMLU-Redux	MMLU-Pro
	(OBQA / ARC)	real	zeros	real	zeros	real / zeros	real / zeros
Qwen3-0.6B	0.527 / 0.637	+12.40	+11.80	+14.68	+15.79	+7.47 / +8.27	+3.52 / +3.83
Qwen3-1.7B	0.991 / 0.998	−6.00	−5.60	−7.00	−6.66	−8.82 / −8.03	−8.66 / −7.95
Qwen3-8B	0.998 / 0.997	−0.20	+0.40	−0.85	−0.43	−0.73 / −0.71	−3.41 / −2.63

checkpoints let us test this without any training: besides 0.6B, they include receivers at 1.7B and 8B, whose accuracy has not been reported before (Table 4). Both larger receivers already spread their answers almost evenly over the options (normalised entropy ≥ 0.99), and on both, C2C lowers accuracy, by 6.0–8.8 points at 1.7B and by 0.2–3.4 points at 8B, while the zero cache stays within 0.8 points of the real one. With no bias to correct, the fuser's constant shift nearly vanishes, and the question-specific part that remains only hurts (Appendix E).

What a trained fuser learns depends on the receiver's bias and the sender's lead. The released pairs in Table 4 match models of similar size, so no sender leads its receiver by more than 6.2 points on OpenBookQA or ARC-C. To vary both factors, we train five fusers with the recipe of the C2C paper's own receiver-scale sweep, pairing receivers from 0.6B to 4B with senders from 0.5B to 7B (Table 5; Appendix I). For each pair, Table 5 lists the receiver's entropy and the sender's lead.

With a biased receiver and a weak sender, the fuser corrects the receiver. With a 0.5B sender, the 0.6B receiver gains more from centring alone (9.4 points on MMLU-Redux) than the sender is ahead of it (3.8 points). The fuser learns the correction: a zero cache keeps 88–105% of the gain across two training seeds, and the fused receiver ends up well above the sender it supposedly learns from (46.5 vs. 38.7 on MMLU-Redux).

Otherwise, the gain depends on the sender. In the other four pairs the zero cache no longer helps: it matches or falls below the receiver alone. On items where the two models disagree, the fused receiver takes the sender's answer on 73–77% with the 7B sender at 0.6B and 1.7B, and on 54–63% with a 3B sender only 4.6 points ahead; at 4B it keeps its own answer more often than it takes the 7B sender's (51–52% vs. 40–44%). Following is lossy: with the 7B sender, the fused 0.6B receiver stays below the sender alone (68.1 vs. 72.1), and a fuser initialised from it still passes only the sender's first digit (Appendix G). Only when the two models are close in accuracy does the fused receiver beat both, as with 1.7B ← 3B and 4B ← 7B on MMLU-Redux (Table 17).

The released recipe stays in the first regime, even with a strong sender. The boundary above holds for fusers trained on task data (MMLU auxiliary-train). The released 0.6B ← Qwen3-4B-Base fuser, trained on general chat data (Teknium, 2023) instead, has a sender 42 points ahead on OpenBookQA. Yet a zero cache keeps 79–91% of its gain (Appendix H), and the fused receiver stays 29 points below the sender alone (52.6 vs. 81.6). No fuser released with C2C follows its sender: each one either corrects its receiver's bias or, where there is none, lowers accuracy.

Table 5: Fusers we train with the C2C paper's task-data recipe (MMLU auxiliary-train). *Entropy*: of the receiver on OpenBookQA; lower means stronger selection bias. *Sender lead*: sender minus receiver alone on MMLU-Redux. Δ: change from the receiver alone with the real cache / zeros. *Same as sender*: share of items where the fused receiver answers like the sender.

			MMLU-Redux			OpenBookQA		
Receiver ← sender	Entropy	Sender lead	Δ real	Δ zeros	Same as sender	Δ real	Δ zeros	Same as sender
0.6B ← 0.5B	0.527	+3.8	+11.59	+10.94	54.7%	+10.40	+9.20	48.0%
1.7B ← 3B	0.991	+4.6	+6.05	−2.58	77.8%	+10.20	−4.80	84.0%
0.6B ← 7B	0.527	+37.2	+33.24	−10.60	81.9%	+41.40	−9.40	84.8%
1.7B ← 7B	0.991	+13.8	+11.22	−24.53	83.9%	+16.40	−25.20	87.8%
4B ← 7B	1.000	+0.7	+3.09	+0.21	81.4%	+5.60	+0.20	87.0%

5 Beyond C2C

RQ1–RQ4 establish the mechanism on C2C. This section asks whether the same explanation applies to other methods of the same kind, and which checks can tell.

5.1 Other methods in this class

The class is defined by three conditions. Wider than the fusers audited above, it includes any method that passes a sender's cache or hidden states to a receiver of about 1B parameters, reports its gain as option accuracy on multiple-choice benchmarks, and reports no control that removes the sender's content. Of 1,307 candidate papers on latent communication (Appendix J), four meet all three: C2C, LCF, XKV and HyLaT (Table 8).

Their receivers show selection bias. It can be measured before any communication, with one forward pass of the receiver alone. Across 21 models from ten families (Appendix B), those that spread their answers evenly over the options (normalised entropy ≥ 0.97) gain at most 2.0 points from centring on OpenBookQA and 0.7 on ARC-C. Qwen3-0.6B, the receiver in the main tables of C2C, LCF and XKV, gains the most of all 21 models (10.9 and 12.1 points). Three of their four receivers show selection bias (entropy 0.31–0.83); only XKV's Llama-3.2-3B does not (0.98).

The one other method we could test behaves like C2C. LCF has released no code, so we reimplemented it from its description (Appendix H). Our version scores 3.5–6.8 points below the published figures, but a zero cache still keeps 75–115% of its gain. XKV has released no code and HyLaT has not released its training data, so we could not test them directly. Some recent work already runs such controls: StateBridge (Peng et al., 2026) compares against a random-noise prefix, MentorPulse (Liu et al., 2026b) against mismatched and random guidance, Interlat (Du et al., 2026) against messages from other tasks and white noise, and two audits of latent communication against another example's message (Cheng et al., 2026; Zhang & Emu, 2026). None of them falls in the class.

Two cheap checks separate transfer from calibration. When a method reports option-accuracy gains on a receiver with selection bias, the gain may come from correcting the receiver rather than from the sender. A *zero-cache control* (Cheng et al., 2026) replaces the sender's cache with zeros and shows how much gain survives without the sender's content. A *recalibrated receiver* subtracts the receiver's mean option logits; it needs no training and beats C2C on two of the four benchmarks. The checks can also favour transfer: with a much stronger sender, the zero cache loses the gain (§4.4).

6 Conclusion

When transfer is the only route to the answer, nothing arrives between the released pair, everything arrives between two Qwen3 models of different size, and a stronger sender of the other family gets across only its verdict (RQ1). Where the released fusers gain, the gain survives deleting the transmitted content (RQ2), is reproduced by one question-independent vector (RQ3), and follows the receiver's selection bias (RQ4). No published main-table pair is on the transfer side of that boundary, and the same one-pass control arm tells the two apart.

ETHICS STATEMENT

This paper audits published methods using their authors' own released weights, code and evaluation protocol. Our disagreement is with the completeness of an evaluation—the absence of an information-free control arm—and not with the integrity of the authors or the value of the architectures they propose: we match the receiver baselines published for the released checkpoint to the second decimal (Table 6), and we take those numbers to be correct. Our conclusions are confined to the regimes in which the gains have so far been measured, and we state where the effect does not hold (§4.4); Appendix J lists the limitations of this study. The study involves no human subjects, no private data and no newly released models; the synthetic tasks we construct contain no personal or sensitive content.

REPRODUCIBILITY STATEMENT

All audited weights are public (`nics-efc/C2C_Fuser`). Our logit scorer is a direct reconstruction of the authors' released routine and is validated by matching published baselines to 0.01 (Table 6). Ablations are driven by a hook on every fuser subclass, with a guard that aborts if the hook count differs from layers $\times$ items—added after an earlier run silently missed a newly added fuser class. Statistical claims use paired bootstrap and Holm-corrected TOST. Task generators, training and evaluation scripts, per-item logits for every arm, and the data of the constructed tasks are provided in an anonymised repository (`https://anonymous.4open.science/r/gains-without-transfer`) and will be released on publication.

AI USE STATEMENT

We used large language models (LLMs) to implement the experiment and analysis code, to run experiments under our direction, to suggest control arms and analyses, to help interpret results, to search for and summarise related work (including the search of Appendix J), to draft parts of the text, and to edit for readability. The research question, the choice of what to report, and the conclusions are the authors'. The authors verified the reported numbers against the per-item outputs in the released repository and revised all LLM-drafted text, and they take full responsibility for the content.

REFERENCES

Siyi Chen, Xiaoyan Zhang, Meng Wu, Jonathan Tremblay, Valts Blukis, Stan Birchfield, Rene Vidal, Alvaro Velasquez, Sijia Liu, and Qing Qu. See what i see, know what i think: Dense latent communication across heterogeneous agents. *arXiv preprint arXiv:2606.13594*, 2026.

Jiaming Cheng, Subhransu Das, and Rajiv Ramnath. When does latent communication pay? a causal audit of relayed kv caches in multi-agent llms. *arXiv preprint arXiv:2608.04893*, 2026.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Lucio M. Dery, Zohar Yahav, Henry Prior, Qixuan Feng, Jiajun Shen, and Arthur Szlam. Latent space communication via k-v cache alignment. *arXiv preprint arXiv:2601.06123*, 2026.

Ricardo Dominguez-Olmedo, Moritz Hardt, and Celestine Mendler-Dünner. Questioning the survey responses of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *International Conference on Machine Learning (ICML)*, pp. 11733–11763, 2024.

Zhuoyun Du, Runze Wang, Huiyu Bai, Zouying Cao, Xiaoyong Zhu, Yu Cheng, Bo Zheng, Wei Chen, and Haochao Ying. Enabling agents to communicate entirely in latent space. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026. arXiv:2511.09149.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.03215.

Aryo Pradipta Gema et al. Are we done with mmlu? *arXiv preprint arXiv:2406.04127*, 2024.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 8048–8057, 2024.

Taekyung Heo, Rasoul Shafipour, Ritchie Zhao, Maximilian Golub, Mohammad Mahdi Kamani, Ritika Borkar, Makesh Tarun Chandran, Pantea Zardoshti, and Bita Darvish Rouhani. Cross-model KV cache transfer in LLM families: A closed-form linear mapping for prefill reuse. *arXiv preprint arXiv:2608.03893*, 2026.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Daniël Lakens. Equivalence tests: A practical primer for t tests, correlations, and meta-analyses. *Social Psychological and Personality Science*, 8(4):355–362, 2017.

Jin-woo Lee, Minkyung Song, Junghyun Oh, Seunghoon Han, Soyoung Park, Gwangseon Jang, and Sungsu Lim. Mixture-of-translators: Translating KV caches across heterogeneous large language models. *arXiv preprint arXiv:2607.28979*, 2026.

Ruizhe Li and Yanjun Gao. Anchored answers: Unravelling positional bias in GPT-2's multiple-choice questions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2439–2465, 2025.

Jiyao Liu, Qi Zhang, Yaoyi Jia, Ziwen Kan, and Song Wang. Dual-cache latent space communication between heterogeneous language models. *arXiv preprint arXiv:2608.20617*, 2026a.

Yuhan Liu, Yuyang Huang, Jiayi Yao, Shaoting Feng, Zhuohan Gu, Kuntai Du, Hanchen Li, Yihua Cheng, Junchen Jiang, Shan Lu, Madan Musuvathi, and Esha Choukse. DroidSpeak: KV cache sharing for cross-LLM communication and multi-LLM serving. *arXiv preprint arXiv:2411.02820*, 2024.

Ziwu Liu, Guozhong Li, Chen Qiu, Weiyang Kong, and Panos Kalnis. MentorPulse: Refreshing cross-model latent guidance for long-form generation. *arXiv preprint arXiv:2608.20927*, 2026b.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Xinyi Mou, Siyuan Wang, Zejun Li, Yulan He, and Zhongyu Wei. HyLaT: Efficient multi-agent communication via hybrid latent-text protocol. *arXiv preprint arXiv:2605.25421*, 2026.

Yanwen Peng, Delvin Ce Zhang, Xi Wang, and Nikolaos Aletras. StateBridge: Training-free hidden-state alignment for latent communication in LLM multi-agent systems. *arXiv preprint arXiv:2608.13317*, 2026.

Pouya Pezeshkpour and Estevam Hruschka. Large language models sensitivity to the order of options in multiple-choice questions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, 2024.

Chau Pham, Boyi Liu, Yingxiang Yang, Zhengyu Chen, Tianyi Liu, Jianbo Yuan, Bryan A. Plummer, Zhaoran Wang, and Hongxia Yang. Let models speak ciphers: Multiagent debate through embeddings. In *International Conference on Learning Representations (ICLR)*, 2024.

Vignav Ramesh and Kenneth Li. Communicating activations between language model agents. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2501.14082.

Yuval Reif and Roy Schwartz. Beyond performance: Quantifying and mitigating label bias in LLMs. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 6784–6798, 2024.

Maximillian Rossi, Prajwal Raghunath, and Eugene Wu. Latent cache flow: Model-to-model communication without text. *arXiv preprint arXiv:2605.22863*, 2026.

Donald J. Schuirmann. A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics*, 15(6):657–680, 1987.

Xi Shi and Qian Lou. KVShareArena: KV-cache reuse across contexts and model checkpoints. *arXiv preprint arXiv:2609.10266*, 2026.

Xiangyu Shi, Marco Chiesa, Gerald Q. Maguire Jr., and Dejan Kostić. KVComm: Enabling efficient LLM communication through selective KV sharing. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.03346.

Teknium. OpenHermes 2.5: An open dataset of synthetic data for generalist LLM assistants. `https://huggingface.co/datasets/teknium/OpenHermes-2.5`, 2023.

Yubo Wang et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *NeurIPS*, 2024.

Sunghyeon Woo, Hoseung Kim, Sunghwan Shim, Minjung Jo, Hyunjoon Jeong, Jeongtae Lee, Joonghoon Kim, Sungjae Lee, Baeseong Park, Se Jung Kwon, and Dongsoo Lee. PrefillShare: A shared prefill module for KV reuse in multi-LLM disaggregated serving. *arXiv preprint arXiv:2602.12029*, 2026.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *Conference on Language Modeling (COLM)*, 2024.

Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. KVLink: Accelerating large language models via efficient KV cache reuse. *arXiv preprint arXiv:2502.16002*, 2025.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. CacheBlend: Fast large language model serving for RAG with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys)*, 2025. arXiv:2405.16444.

Hancheng Ye, Zhengqi Gao, Mingyuan Ma, Qinsi Wang, Yuzhe Fu, Ming-Yu Chung, Yueqian Lin, Zhijian Liu, Jianyi Zhang, Danyang Zhuo, and Yiran Chen. KVCOMM: Online cross-context KV-cache communication for efficient LLM-based multi-agent systems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2510.12872.

Huixiang Zhang and Mahzabeen Emu. Do latent channels actually communicate? a causal audit of latent multi-agent LLM communication. *arXiv preprint arXiv:2607.26773*, 2026.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning (ICML)*, 2021.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. Large language models are not robust multiple choice selectors. In *International Conference on Learning Representations (ICLR)*, 2024.

Han Zhou, Xingchen Wan, Lev Proleev, Diana Mincu, Jilin Chen, Katherine Heller, and Subhrajit Roy. Batch calibration: Rethinking calibration for in-context learning and prompt engineering. In *International Conference on Learning Representations (ICLR)*, 2024.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2511.20639.

A PROTOCOL ALIGNMENT

Table 6 compares our reproduction with the values that Rossi et al. (2026) publish for the released 0.6B checkpoint under the logit protocol, and with the C2C paper's own free-form figures (Fu et al., 2026).

Table 6: Protocol alignment against the values that Rossi et al. (2026) publish for the released 0.6B checkpoint under the logit protocol. Two receiver baselines and the C2C ARC-C number match to within 0.01; all four oracle-frontier (ORF) values match to within 0.7. Remaining gaps trace to undisclosed details (Rossi et al. (2026) report 1150 ARC-C items; the test split has 1172, of which 1165 are 4-option). The last two rows give the C2C paper's own figures (Fu et al., 2026), obtained by free-form decoding and answer extraction, which is why they differ from both columns.

Quantity	Ours	Rossi et al. (2026)	Δ
Receiver, OpenBookQA	**39.20**	39.20	0.00
Receiver, MMLU-Redux	**34.91**	34.91	0.00
Receiver, ARC-C	39.33	39.91	-0.58
Receiver, MMLU-Pro	15.58	16.17	-0.59
C2C, ARC-C	**54.01**	54.00	$+0.01$
C2C, MMLU-Redux	42.38	42.54	-0.16
C2C, OpenBookQA	51.60	50.80	$+0.80$
ORF (OBQA / ARC-C / MMLU-R / MMLU-Pro)	56.00 / 54.10 / 48.97 / 20.89	55.40 / 54.43 / 48.90 / 20.95	< 0.7
Quantity	Ours	Fu et al. (2026), free-form	Δ
Receiver, OBQA / ARC-C / MMLU-R	39.20 / 39.33 / 34.91	39.20 / 41.04 / 35.53	0.00 / -1.71 / -0.62
C2C, OBQA / ARC-C / MMLU-R	51.60 / 54.01 / 42.38	52.60 / 54.52 / 42.92	-1.00 / -0.51 / -0.54

B SELECTION BIAS ACROSS SMALL MODELS

Table 7 reports, for each model, the accuracy, the share of items assigned to its single most-favoured option, that option, the normalised entropy of its prediction distribution, and the centring gain, under exactly the logit protocol of §3. The centring gain is the two-fold cross-fitted accuracy gain from centring in §4.3 (20 random splits; standard deviation ≤ 0.6 points), computed by the same routine as the Centring column of Table 3, which it reproduces for Qwen3-0.6B to within 0.05. The first thirteen models were fixed before running; Llama-3.2-3B was added to check the third receiver of XKV, and seven further models were added later, mostly to fill the 1–4B range and to include the senders of Table 5. Two base models without a chat template (OPT-350M, Pythia-410M) fall back to a plain-text prompt.

Table 8 lists the methods in the class of §5, with the entropy of the receivers in their main tables.

C AGREEMENT BETWEEN ARMS AND CROSS-FITTED CALIBRATION

Table 9 compares the real and zeroed arms item by item at receiver 0.6B. Displacements are row-centred option-logit differences from the receiver alone, Δ_{real} and Δ_{ZERO}. Table 10 estimates the option-mean bias on one four-option benchmark and applies it to another, restricted to four-option items.

D ENSEMBLES OF THE TWO BASE MODELS

Table 11 reports the reference lines around the per-item oracle of §4.2: the two base models, the oracle routing frontier (ORF, $P(\text{receiver correct} \cup \text{sender correct})$), C2C, the blank page, and three ways of pooling the two models' beliefs (averaging option probabilities, adding log-probabilities, adding logits). "Calibrated" centres each model's logits with held-out option means before averaging probabilities (two-fold cross-fitted). "Both wrong" is the share of items neither base model answers correctly; "rescued" is the share of those items the probability-averaging ensemble answers correctly. Pooling is correct on 0.0–0.4% of the items where both models are wrong, so the loophole by which a probability-transferring method could legitimately cross the oracle line is small; the ensemble becomes competitive only once each model is centred first, at which point it too crosses the line on MMLU-Pro.

Table 7: Selection bias and centring gain by scale. Entropy is normalised to $[0, 1]$, where 1 is uniform over the options; Centr. is the centring gain in accuracy points. Degeneracy is common below 1B and rare above 2B; it is necessary for a centring gain but not sufficient: the five models that always answer one letter have little to recover (at most 3.5), every other degenerate model gains 0.9–10.9 / 2.7–12.1 points, no model with entropy above 0.97 gains more than 2.0 / 0.7, and Qwen3-0.6B has by far the largest centring gain.

		OpenBookQA					ARC-Challenge				
Model	Family	Acc	Top%	Opt	Ent	Centr.	Acc	Top%	Opt	Ent	Centr.
SmolLM2-135M-Instruct	SmolLM	27.60	100.0	A	**0.000**	−5.2	22.70	100.0	A	**0.000**	+2.9
OPT-350M†	OPT	27.60	100.0	A	**0.000**	−7.5	22.70	100.0	A	**0.000**	+3.5
SmolLM2-360M-Instruct	SmolLM	27.40	99.8	A	0.010	+1.8	22.70	100.0	A	**0.000**	+2.2
Pythia-410M†	Pythia	27.60	100.0	A	**0.000**	−3.1	22.70	100.0	A	**0.000**	+1.2
Qwen2-0.5B-Instruct§	Qwen2	42.60	41.6	A	0.801	+0.9	36.26	45.2	A	0.763	+2.7
Qwen2.5-0.5B-Instruct	Qwen2.5	45.40	61.8	A	0.741	+3.6	41.21	55.8	A	0.831	+8.3
Qwen3-0.6B	Qwen3	39.20	79.0	A	0.527	+10.9	39.33	72.2	A	0.637	+12.1
Gemma-3-1B-it	Gemma	34.40	89.8	A	0.307	+1.6	28.75	87.8	A	0.364	+3.4
Llama-3.2-1B-Instruct	Llama	44.80	42.0	A	0.828	+3.0	41.21	46.3	B	0.861	+5.6
TinyLlama-1.1B-Chat	TinyLlama	27.60	100.0	A	**0.000**	+0.0	22.70	100.0	A	**0.000**	+0.4
Qwen2.5-1.5B-Instruct	Qwen2.5	64.20	35.2	D	0.981	+2.0	71.93	34.0	D	0.983	−0.6
Qwen2.5-Math-1.5B-Inst.§	Qwen2.5	42.40	32.0	C	0.973	−0.0	47.53	32.6	D	0.973	+0.3
Qwen3-1.7B	Qwen3	66.00	32.2	A	0.991	−0.7	76.45	27.0	B	0.998	−0.6
SmolLM2-1.7B-Instruct§	SmolLM	50.80	60.2	A	0.713	+3.1	48.98	52.6	A	0.813	+6.4
Gemma-2-2B-it§	Gemma	68.20	35.2	A	0.977	+1.5	73.21	28.1	A	0.996	+0.3
Qwen2.5-3B-Instruct§	Qwen2.5	73.80	37.8	A	0.970	+1.3	80.63	27.8	A	0.997	−0.1
Llama-3.2-3B-Instruct‡	Llama	71.20	33.6	A	0.984	+0.1	72.27	28.1	B	0.996	+0.5
Phi-4-mini (3.8B)§	Phi	79.80	36.0	A	0.972	+1.5	81.83	27.9	A	0.994	+0.7
Qwen3-4B	Qwen3	79.40	26.0	B	1.000	−0.9	87.37	26.8	C	0.998	+0.1
Qwen2.5-7B-Instruct§	Qwen2.5	85.80	29.4	A	0.993	−0.2	90.36	26.5	C	0.998	−0.6
Qwen3-8B	Qwen3	84.80	26.8	B	0.998	+1.2	91.30	27.5	B	0.997	+0.0

† Base model without a chat template; evaluated with a plain-text prompt. Its accuracy equals the share of items whose gold answer is A, which is what a constant prediction of A produces. ‡ Added after the list was fixed, to check the third receiver of XKV in Table 8. § Added later, mostly to fill the 1–4B range or as a sender of Table 5.

Table 8: Methods in the class of §5: cross-model latent-communication methods that report gains as option accuracy with a receiver of about 1B parameters. Entropy: normalised entropy of the receiver's answers on OpenBookQA (Appendix B); lower means stronger selection bias.

Work	Receiver(s) in main table	Entropy	Information-free control arm
C2C (Fu et al., 2026)	Qwen3-0.6B	0.527	not reported (Identical ablation only, §4.2)
LCF (Rossi et al., 2026)	Qwen3-0.6B	0.527	not reported
XKV (Liu et al., 2026a)	Qwen3-0.6B / Gemma-3-1B Llama-3.2-3B	0.527 / 0.307 0.984	not reported[a]
HyLaT (Mou et al., 2026)	Llama-3.2-1B	0.828	not reported[b]

[a] No receiver-alone baseline either. [b] Gaussian-noise robustness test only; no replacement arm.

E Decomposition at 1.7B and 8B

Table 12 repeats the analysis of §4.3 on the released 1.7B and 8B fusers. Cross-fitted columns as in Table 3.

At 8B the fuser mostly rescales the receiver's logits. On OpenBookQA and ARC-C, -0.86 times the receiver's row-centred logits explains 99.6% of the displacement energy, and the mean gap between the two highest options shrinks from 15.4 to 2.1 and from 19.7 to 2.6. The constant μ carries only 3.6–6.0% of the energy, but its norm (3.1 and 4.2) exceeds that gap, so adding E alone, which equals removing μ from the fused logits, flips many predictions. The low "Only E" accuracies at 8B therefore reflect this rescaling rather than harmful question-specific content.

Table 9: Per-item agreement between the real cache and the blank page. "Same change": among items on which the real cache changes the receiver's prediction, the share on which ZERO makes the same change.

Benchmark	n	real=ZERO	real=SHUFFLE	same change	relative residual energy	median cos
OpenBookQA	500	94.6%	95.4%	92.0%	0.4%	0.999
ARC-C	1172	95.4%	94.9%	92.9%	0.5%	0.999
MMLU-Redux	5632	94.2%	94.4%	91.4%	0.4%	0.999
MMLU-Pro	12032	92.1%	91.6%	85.9%	0.6%	0.998

Table 10: Cross-benchmark centring (four-option items). Rows: benchmark on which the option means are estimated; columns: benchmark on which they are applied. Receiver alone on the same items: 39.20 / 39.14 / 34.91.

Bias estimated on	OpenBookQA	ARC-C	MMLU-Redux
OpenBookQA	50.20	52.96	44.32
ARC-C	49.60	51.59	44.35
MMLU-Redux	49.40	52.36	44.32

F EQUIVALENCE TESTS

Table 13 lists the 20 real-versus-ZERO comparisons of §4.2: the point estimate (real minus ZERO, in points), the 95% paired-bootstrap interval, and the Holm-corrected TOST p-value at $\delta = \pm 2$ points (Schuirmann, 1987; Lakens, 2017; Holm, 1979).

G CONSTRUCTED TASKS: FULL TABLES

Tables 14 and 15 give the complete results of the two constructed tasks of §4.1, and Table 16 the per-digit results for the further senders, including the training variants omitted from Table 1: initialising the cross-model fuser from the released 0.6B fuser rather than from scratch, tripling the budget to 3,000 steps, and training under the public recipe with gate noise rather than with the gates fixed open. The fuser is the released architecture with the released layer mapping (28 fusers, receiver layer t fed from sender layer $\max(0, t-4)$; 478M parameters), trained with the public recipe (AdamW, 10^{-4}, 1,000 steps of 32 sequences, Gumbel-sigmoid gates annealed from 1.0 to 0.001), both backbones frozen. Under this recipe the gate logits never leave ± 0.03, a coin flip per position, which is why the main runs fix the gates open. Train and test items draw their record names from disjoint pools (40,000 and 2,000 items) and use eight sentence templates; the receiver's placeholder `0000` has the length of every code and aligns token by token under both tokenizers on all items. Exact match is read from one teacher-forced pass: an item counts when the most likely digit at each of the four answer positions is the gold digit, which is exactly when greedy decoding outputs the gold code. In the mismatched arm, the receiver answers with one of the eight codes stored in the other item's records on 17.2% of items with the same-model fuser and on 19.0% with the Qwen3-1.7B sender, against 0.05% with the Qwen2.5-7B sender and 0.08% by chance.

H ROBUSTNESS OF THE CONTENT ABLATION

This appendix gives the full account of the robustness checks summarised in §4.2.

The conclusion is not confined to one sender, one fuser architecture or one decoding protocol. To test whether the phenomenon is an artifact of one model family, we hold the receiver at Qwen3-0.6B and vary the sender—including Llama-3.2-1B-Instruct, which requires dual-tokenizer alignment—and ZERO retains 79–120% of the gain in all six cells (95/108% for Qwen2.5-0.5B, 120/96% for Llama-3.2-1B, 79/91% for Qwen3-4B-Base).

This last sender also corrects a reported anomaly. The C2C paper notes that Qwen3-4B-Base scores only 2.20% alone yet is the strongest sender, concluding that a sender's standalone accuracy does not predict its usefulness. Under the logit protocol the same model scores 81.60/87.71. The anomaly

Table 11: Ensemble reference lines at receiver 0.6B, sender Qwen2.5-0.5B-Instruct.

Benchmark	Recv.	Sender	ORF	C2C	ZERO	Prob. avg	Log-prob	Logit	Calibrated	Both wrong	Rescued
OpenBookQA	39.20	45.40	56.00	51.60	51.00	41.40	41.00	41.20	52.68	44.0%	0.0%
ARC-C	39.33	41.21	54.10	54.01	55.12	41.30	41.64	41.38	53.04	45.9%	0.4%
MMLU-Redux	34.91	38.69	48.97	42.38	43.18	36.12	36.03	35.87	45.81	51.0%	0.1%
MMLU-Pro	15.58	15.50	20.89	19.10	19.41	15.45	14.88	14.84	24.21	79.1%	0.0%

Table 12: Displacement decomposition and centring at 1.7B and 8B.

Receiver	Benchmark	Recv.	C2C	ZERO	const. energy	median cos	Centring	Only μ	Only E	real=ZERO
1.7B	OpenBookQA	66.00	60.00	60.40	0.3%	-0.08	65.30	65.42	60.20	96.2%
1.7B	ARC-C	76.45	69.45	69.80	1.2%	0.38	75.96	75.93	65.78	96.6%
1.7B	MMLU-Redux	58.35	49.52	50.32	0.7%	0.12	58.24	57.95	49.85	95.5%
1.7B	MMLU-Pro	30.48	21.82	22.53	13.9%	0.47	31.92	31.29	16.91	95.4%
8B	OpenBookQA	84.80	84.60	85.20	3.6%	0.31	86.09	86.18	61.39	97.2%
8B	ARC-C	91.30	90.44	90.87	5.0%	0.40	91.34	91.47	69.35	98.1%
8B	MMLU-Redux	75.64	74.91	74.93	5.1%	0.42	75.56	75.49	59.68	97.2%
8B	MMLU-Pro	47.02	43.62	44.40	6.0%	0.29	47.95	47.74	27.95	93.5%

is an artefact of free-form formatting, not a property of representations. The instructive fact is the opposite one: a sender that scores 81.60 lifts the receiver only to 52.60, of which 79% is reproduced by a blank page.

The finding does not depend on the logit protocol either. With a fuser we trained ourselves (Gemma-3-1B-it sender, Qwen3-0.6B receiver, 573 steps on OpenHermes under the public recipe) and free-form greedy decoding, the blank page scores 51.60 against 47.00 for the real cache on OpenBookQA, and 40.13 against 38.67 on MMLU-Redux.

A second question is whether the conclusion survives a different fuser architecture. We reimplemented LCF from its description (no code is released); at d=128 our implementation has 19,389,944 trainable parameters against the stated 19.4M. Trained once under the public recipe it reaches 45.80/52.56/40.98/16.42, i.e. 3.5–6.8 points below the published figures, a gap we cannot close without code. On our own checkpoint ZERO retains 75–115% of the gain (mean 99.5%). This is an existence statement, not a verdict on the published numbers: as far as we can reproduce it, the effect is not specific to the C2C architecture. That our absolute numbers are lower does not weaken this: whatever we failed to reproduce must first clear the bar that a blank page already reaches.

I FUSERS TRAINED UNDER THE IN-DISTRIBUTION RECIPE

Table 17 gives every arm behind Table 5. All five fusers use the released C2C architecture and layer mapping and the public recipe of the C2C paper's Figure 6 sweep: the first 40,000 items of the MMLU auxiliary-train split in chat format (39,867 after the 1024-token filter), one epoch, AdamW at 10^{-4} with linear decay and 10% warm-up, effective batch 32, both backbones frozen. Senders are the instruction-tuned Qwen2.5 models; all three arms and both baselines use the logit scorer of §3. The centring gains cited in §4.4 are the two-fold cross-fitted centring gains of §4.3 computed on the receiver's own logits (20 splits): $+9.40$ / $+10.93$ at 0.6B, -0.10 / -0.59 at 1.7B and -0.49 / -0.85 at 4B on MMLU-Redux / OpenBookQA. The 0.6B ← 0.5B run used micro-batches of 8 with 4 accumulation steps, the others micro-batches of 4 with 8; the effective batch, number of optimiser steps and schedule are identical. Each fuser was trained once, except the two pairs that anchor the two regimes, which we retrained with a second seed. For 0.6B ← 0.5B, real / ZERO / SHUFFLE reach 47.30 / 47.64 / 47.23 on MMLU-Redux and 52.40 / 53.00 / 52.80 on OpenBookQA, against 46.50 / 45.85 / 46.11 and 49.60 / 48.40 / 50.20 for the first seed; the blank page retains 103–105% of the gain under the second seed and 88–94% under the first. For 1.7B ← 3B, the second seed gives 64.56 / 56.55 / 52.18 and 78.20 / 62.80 / 59.40 against 64.40 / 55.77 / 51.47 and 76.20 / 61.20 / 55.80: every arm moves by at most 3.6 points, and the blank page again falls below the receiver alone on MMLU-Redux (58.35) and OpenBookQA (66.00).

Table 13: TOST equivalence cells. A cell is equivalent when Holm-adjusted $p < 0.05$.

Fuser / receiver	Benchmark	n	real − ZERO	95% CI	Holm p	Equivalent
C2C 0.6B	OpenBookQA	500	+0.60	[−1.00, +2.20]	0.274	no
C2C 0.6B	ARC-C	1172	−1.11	[−2.13, −0.09]	0.252	no
C2C 0.6B	MMLU-Redux	5632	−0.80	[−1.31, −0.30]	< 0.001	yes
C2C 0.6B	MMLU-Pro	12032	−0.32	[−0.58, −0.05]	< 0.001	yes
C2C 1.7B	OpenBookQA	500	−0.40	[−2.00, +1.00]	0.237	no
C2C 1.7B	ARC-C	1172	−0.34	[−1.19, +0.51]	0.001	yes
C2C 1.7B	MMLU-Redux	5632	−0.80	[−1.24, −0.36]	< 0.001	yes
C2C 1.7B	MMLU-Pro	12032	−0.71	[−0.96, −0.48]	< 0.001	yes
C2C 8B	OpenBookQA	500	−0.60	[−2.00, +0.60]	0.218	no
C2C 8B	ARC-C	1172	−0.43	[−1.11, +0.26]	< 0.001	yes
C2C 8B	MMLU-Redux	5632	−0.02	[−0.39, +0.36]	< 0.001	yes
C2C 8B	MMLU-Pro	12032	−0.78	[−1.09, −0.47]	< 0.001	yes
LCF (ours) 0.6B	OpenBookQA	500	−1.00	[−3.00, +1.00]	0.785	no
LCF (ours) 0.6B	ARC-C	1172	+0.51	[−0.60, +1.71]	0.042	yes
LCF (ours) 0.6B	MMLU-Redux	5632	−0.75	[−1.30, −0.21]	< 0.001	yes
LCF (ours) 0.6B	MMLU-Pro	12032	+0.21	[0.00, +0.42]	< 0.001	yes
C2C, Llama-3.2-1B sender	OpenBookQA	500	−2.00	[−4.40, +0.40]	1.000	no
C2C, Llama-3.2-1B sender	ARC-C	1172	+0.68	[−0.85, +2.22]	0.265	no
C2C, Qwen3-4B-Base sender	OpenBookQA	500	+2.80	[−0.40, +6.00]	0.707	no
C2C, Qwen3-4B-Base sender	ARC-C	1172	+1.62	[−0.26, +3.50]	1.000	no

Table 14: 4-digit code repetition task, n=2,000. Exact match, mean gold-digit probability (chance 0.10) and mean digit entropy (uniform = 2.30). Cross-model: Qwen2.5-0.5B → Qwen3-0.6B; same-model: Qwen3-0.6B → Qwen3-0.6B, identical code, data and recipe. Control arms: mismatched / ZERO / SHUFFLE.

	Condition	Train loss	Exact (%)	p(gold)	Entropy
Reference	Receiver, masked (floor)	—	0.00	0.101	1.19
	Receiver, unmasked (ceiling)	—	85.60	0.913	0.26
	Sender, unmasked	—	78.80	0.878	0.34
Cross-model fuser	from scratch, real cache	2.30	0.00	0.100	2.30
	control arms		0.00 / 0.00 / 0.00	0.100	2.30
	initialised from released fuser	2.30	0.05	0.100	2.30
	3,000 steps	2.30	0.00	0.100	2.30
	gates fixed open	2.30	0.00	0.100	2.30
Same-model fuser	public recipe, real cache	0.15	92.25	0.957	0.07
	gates fixed open, real cache	0.05	**95.45**	0.974	0.05
	control arms		0.00 / 0.00 / 0.05	0.101	0.68 / 1.94 / 0.97
Pipeline control	Receiver also unmasked	0.003	—	—	—

Table 5's "same as sender" includes items on which both models are right. Restricted to items on which the receiver alone and the sender alone disagree, the fused prediction takes the sender's answer / keeps the receiver's on 26.8 / 52.8% and 26.3 / 48.9% of them for 0.6B ← 0.5B (MMLU-Redux / OpenBookQA), 53.6 / 33.6% and 63.0 / 27.8% for 1.7B ← 3B, 76.8 / 15.7% and 76.5 / 17.1% for 0.6B ← 7B, 72.6 / 16.3% and 73.9 / 13.7% for 1.7B ← 7B, and 39.7 / 52.3% and 43.8 / 51.0% for 4B ← 7B (first seed of each pair).

J LIMITATIONS AND LITERATURE SEARCH

All benchmark results are option accuracies from one logit scorer; free-form generation is checked only once (Appendix H). The released fusers all have Qwen3 receivers, and other families enter only through fusers we trained or reimplemented. Of the four methods in the class of §5, only C2C could be tested with its released weights: our LCF reimplementation scores 3.5–6.8 points below its published figures, and XKV and HyLaT were not tested. Most fusers we trained were trained once; the two pairs that anchor the regimes of §4.4 have two seeds.

Table 15: Colour lookup task, n=2,000, four options, chance 25%. Accuracy, mean normalised probability of the gold option (chance 0.25) and entropy over the four options (uniform $= \ln 4 =$ 1.386). Model pair and training recipe as in Table 14; control arms: mismatched / ZERO / SHUFFLE.

	Condition	Train loss	Accuracy (%)	p(gold)	Entropy
Reference	Receiver, masked (floor)	—	25.70	0.252	1.127
	Receiver, unmasked (ceiling)	—	64.90	0.509	0.940
	Sender, unmasked	—	27.80	0.259	1.264
Cross-model fuser	gates fixed open, real cache	1.385	25.35	0.250	1.385
	control arms		25.46 / 25.05 / 25.60	0.250	1.385
	public recipe, real cache	1.389	25.55	0.249	1.376
	control arms		25.46 / 25.45 / 25.40	0.249	1.376
Same-model fuser	gates fixed open, real cache	0.264	**90.25**	0.875	0.210
	control arms		23.21 / 24.90 / 25.65	0.237 / 0.250 / 0.253	0.346 / 0.795 / 0.688
	second training run, real cache / control arms	0.201	90.55 / 23.71 / 24.90 / 24.55	0.873 / 0.239 / 0.254 / 0.246	0.221 / 0.362 / 0.826 / 0.745
Stronger sender Qwen2.5-7B-Inst.	sender alone, unmasked	—	97.30	0.969	0.024
	gates fixed open, real cache	0.031	**97.85**	0.968	0.055
	control arms		23.71 / 26.50 / 25.75	0.238 / 0.254 / 0.254	0.162 / 1.066 / 0.578

Table 16: Three further senders for the 4-digit code repetition task (gates fixed open, 1,000 steps, otherwise the recipe of Table 14; the 3B sender has 36 layers, mapped to the receiver's 28 by aligning the last layers): accuracy of each of the four answer digits against the queried item (%), exact match and digit entropy. "init. from §4.4 fuser" initialises the fuser from the content-carrying 0.6B ← 7B fuser of Table 5. In the mismatched arm of the 7B sender, the first digit equals the *other* item's first digit on 98.9% of items.

Sender	Arm	Train loss	Digit 1	Digit 2	Digit 3	Digit 4	Exact (%)	Entropy
Qwen3-1.7B	sender alone, unmasked	—	94.0	98.8	99.9	100.0	93.20	0.06
	real cache	0.021	98.4	99.4	99.4	99.2	**97.95**	0.01
	mismatched		9.6	10.1	10.7	9.8	0.00	0.61
	ZERO		9.4	10.6	10.3	9.3	0.05	1.80
	SHUFFLE		18.3	19.1	17.9	17.1	0.00	1.04
Qwen2.5-3B-Instruct	sender alone, unmasked	—	98.4	99.7	100.0	100.0	98.10	0.01
	real cache	1.88	**80.4**	9.8	10.6	10.4	0.00	1.91
	mismatched		10.4	9.9	10.3	9.1	0.00	1.92
	ZERO		10.1	11.6	9.5	10.5	0.05	2.09
	SHUFFLE		9.7	9.7	10.0	10.9	0.00	1.98
Qwen2.5-7B-Instruct	sender alone, unmasked	—	99.8	100.0	100.0	100.0	99.75	0.00
	real cache	1.73	**98.9**	9.9	9.8	10.3	0.00	1.73
	real cache, init. from §4.4 fuser	1.73	98.7	10.9	9.5	10.0	0.00	1.74
	mismatched		9.7	10.1	9.4	9.8	0.00	1.73
	ZERO		11.1	11.6	10.6	9.5	0.00	2.13
	SHUFFLE		12.4	9.4	10.0	9.4	0.00	1.99

Two factors are not separated. A biased receiver exists only at 0.6B, so selection bias and receiver size are not varied independently; and the fusers of §4.4 are trained on MMLU auxiliary-train, which is in distribution for MMLU-Redux, whereas the released fusers are trained on general chat data. In RQ1 we attribute the failure of the released pair to representation mismatch without probing the sender's cache directly. On the benchmarks, the shuffled cache keeps the question's tokens, and we did not run an arm that substitutes another item's cache, the primary control of Cheng et al. (2026); the zero and noise arms carry no information about the question.

The candidate papers of §5 were collected on 7 September 2026 from 40 keyword queries to the arXiv API (1,202 papers), the Semantic Scholar citation graphs of seven latent-communication papers (221) and two curated lists (40), 1,307 after removing duplicates. The 193 candidates with the highest relevance scores were screened by title, 45 were read in full, and four meet the three conditions of §5.

Table 17: All arms for the fusers of Table 5. Accuracy (%); "=recv" and "=sender" give the share of items on which the fused prediction equals the receiver's own or the sender's own prediction.

Receiver ← sender	Benchmark	Receiver	Sender	C2C	ZERO	SHUFFLE	=recv	=sender
0.6B ← 0.5B	MMLU-Redux	34.91	38.69	46.50	45.85	46.11	64.3%	54.7%
	OpenBookQA	39.20	45.40	49.60	48.40	50.20	56.6%	48.0%
1.7B ← 3B	MMLU-Redux	58.35	62.93	64.40	55.77	51.47	70.4%	77.8%
	OpenBookQA	66.00	74.00	76.20	61.20	55.80	72.6%	84.0%
0.6B ← 7B	MMLU-Redux	34.91	72.12	68.15	24.31	27.15	44.1%	81.9%
	OpenBookQA	39.20	85.80	80.60	29.80	30.80	49.4%	84.8%
1.7B ← 7B	MMLU-Redux	58.35	72.12	69.57	33.82	29.33	63.2%	83.9%
	OpenBookQA	66.00	85.80	82.40	40.80	30.40	68.4%	87.8%
4B ← 7B	MMLU-Redux	71.45	72.12	74.54	71.66	70.60	84.7%	81.4%
	OpenBookQA	79.40	85.80	85.00	79.60	79.00	88.4%	87.0%