ONE STRING PER TASK: HOW A BENCHMARK'S FIXED INSTRUCTIONS CAN BECOME TASK-ID TOKENS

Anonymous authors
Paper under double-blind review

ABSTRACT

Universal multimodal embedding benchmarks pair each task with one fixed instruction string. This makes a familiar wording useful as a task cue, while leaving its contribution to the score unmeasured. We expose this blind spot by changing only the query instruction and holding the image, query content, and candidate index fixed. Across 36 MMEB tasks, a meaning-preserving rewrite costs VLM2Vec-Qwen2VL-2B 3.3 accuracy points on average, compared with 10.0 for a wrong-task instruction. The ordering replicates on three further checkpoints, with sensitivity varying by training recipe. Training controls explain why the distinction matters: a model given meaningless task tokens nearly matches fixed-instruction training, and randomly assigning which wording is trained reverses the model's wording preference. These results support learned reliance on the exact string alongside instruction meaning. Sampling paraphrases during training more than halves the gap on held-out wordings. We release a benchmark-agnostic audit and recommend reporting paraphrase and wrong-instruction gaps beside every fixed-wording score, so that evaluation measures how much performance survives an equivalent instruction.

1 INTRODUCTION

Universal multimodal embedders (UMEs) compress an image, a text query, or an interleaved instruction-conditioned query into a single dense vector, compared against a candidate pool with cosine similarity. Because one encoder must serve retrieval, classification, VQA, and grounding at once, most UMEs built since Jiang et al. (2025) condition their output on a natural-language *instruction* describing the current task.

The convention, and its blind spot. The field's dominant evaluation, MMEB (Jiang et al., 2025), pairs each of its 36 image tasks with exactly one instruction string (23 distinct strings in all). Every query in a task receives the same string. This is a reasonable simplification for measuring raw task accuracy, but not for measuring *instruction-following*. An instruction does two jobs at once. It carries meaning the model parses, and, because it never varies within a task, it also carries a task-constant surface form the model can key on instead. A model can therefore score well by learning which string maps to which response format, without parsing what the string asks. MMEB cannot detect the difference, because the string never varies and every query is scored only against its own task's candidates. A model whose score barely moves when the instruction is deleted is scored exactly like one that depends on it. That models are sensitive to wording is well documented (§2). What we add is the protocol property that hides this sensitivity, together with a way to measure it.

The audit. We make the blind spot measurable by varying what MMEB holds fixed. Each of the 36 tasks is scored under six conditions: its own instruction, three author-curated meaning-preserving paraphrases, another task's instruction (semantically wrong here), and no instruction. Only the query's instruction changes. The image, the query content and the once-encoded candidate pool stay as MMEB built them (Figure 1b). The paraphrase gap thus measures what a rewording costs against the benchmark's own index, and the wrong-instruction gap is the yardstick: for a genuine instruction-follower it should far exceed the paraphrase gap.

Three findings. First, *equivalent wording is not equivalent to the model*. A paraphrase costs about a third as much as a wrong-task instruction on VLM2Vec-Qwen2VL-2B (Figure 1b). The ordering replicates across checkpoints, although its magnitude depends on the training recipe (§3). Second, *the*

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.

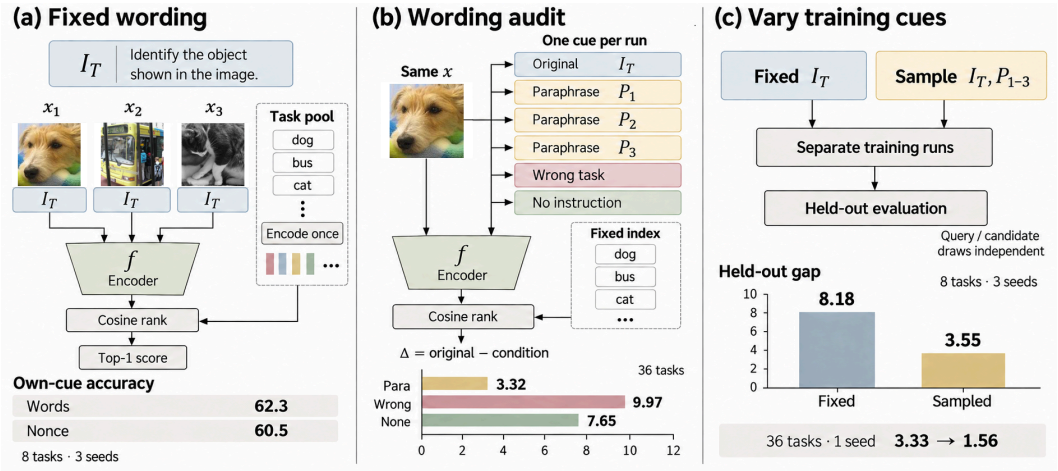


Figure 1: **The fixed-string blind spot.** **(a)** One instruction per task carries both meaning and a constant cue; a nonce-trained model nearly matches fixed-wording training (8 tasks, three seeds, each given its trained cue). **(b)** Our six-condition audit varies the query instruction against a fixed candidate index; bars show mean costs for VLM2Vec-Qwen2VL-2B over 36 tasks (300 queries each; paraphrases averaged within task). **(c)** Sampling paraphrases during training more than halves the mean held-out gap (8 tasks, three seeds; the bottom row reports 36 tasks at one seed; §6). Square-cropped images and candidate labels are from the MMEB VOC2007 evaluation split; image-token syntax is omitted. Bars report task-aggregated results.

score rewards a stable cue even without meaningful words. A model fine-tuned with a meaningless task token nearly matches the same recipe trained on the real instructions (§4). Third, *wording preference follows training exposure*. When a lottery chooses which wording is trained, the trained wording wins within task type (§4). Varying wording during training then reduces the cost on new paraphrases (§6).

These results hold at three levels of strength. That rewording moves the score, and that the protocol cannot see it, are direct measurements and our most solid contribution. That training exposure creates a preference for a wording is supported by the randomized lottery. That a natural-language checkpoint uses its string as a *task-ID token*, in the vocabulary of shortcut learning (Song et al., 2024), is the reading most consistent with the results. A model can learn both what an instruction means and which familiar string identifies its task; fixed-wording evaluation leaves these contributions entangled. We use *template sensitivity* for the measured effect throughout.

Contributions.

- **An audit of the fixed-string blind spot.** Six instruction conditions over 36 MMEB tasks and five checkpoints measure sensitivity to equivalent and wrong-task wording against the benchmark's own index (§3).
- **Evidence about what drives the cost.** Randomized wording exposure and a nonce control connect the effect to training, while query-only, candidate-only, and joint rewrites test the template-alignment explanation (§4–5).
- **A training intervention and a reporting tool.** Paraphrase augmentation more than halves the held-out gap. The released audit reports paraphrase and wrong-instruction gaps beside fixed-wording accuracy, making this dependence visible (§6).

2 Related Work

A search of 95 full-text 2025–26 UME papers finds no prior audit of instruction-paraphrase robustness on MMEB: five mention instruction robustness and one tests it in remote sensing (Appendix B.24).

Nearest work, and how it differs. Musacchio et al. (2026) trace embedding sensitivity to sequence-end tokens and pooling, a token-format effect rather than an instruction-content one. Tonmoy et al. (2026) report a 25.6-point R@10 swing between bare-input and chat-template formatting, again a format effect. Sánchez Aimar et al. (2025) train with ten instruction templates per task in remote sensing and find it cuts the loss on unseen rewordings from 10.05 to 3.53 points, the nearest precedent for Section 6; they do not audit how a fixed-string protocol scores the single-template model. Huang et al. (2026) diagnose a failure to enforce the instructed *target modality*, a distinct axis: a model can infer that a query wants an image back while still scoring "identify the object" and "name the object" differently. Li et al. (2026b) show that internal layers hold signal the pooled vector discards; our adapted layerwise probe finds target type no more decodable inside the stack than at the final layer (Appendix B.30). Li et al. (2026c) is the nearest genre match, a broad study of universal multimodal retrieval recipes, and Attimonelli et al. (2026) the nearest methodological match, an audit of composed image retrieval benchmarks for unimodal shortcuts. We target one protocol property of MMEB, a fixed instruction string per task. We perturb the query, the candidate and both sides, add a nonce-trained control that shows what the protocol cannot detect, and vary the property in training.

Prompt variation at evaluation time. Instruction-conditioned retrieval goes back to Asai et al. (2023). The case for varying the prompt at evaluation time is made for LLMs by Mizrahi et al. (2024) and Romanou et al. (2026), and for text embeddings by Kostiuk & Enevoldsen (2026) (6 models, 11 datasets, 38–45 prompts per task) and by Frank & Afli (2025), who paraphrase inputs stochastically at evaluation time. Both text-embedding studies vary far more wordings than our three per task. Our setting differs from theirs in three ways. First, MMEB conditions queries *and* candidates on instruction templates and scores against a once-encoded candidate index, so a query-side rewrite can fail from misalignment with that frozen index rather than from lost legibility; we separate the two by perturbing each side and both (Section 5). Second, the *swap* control turns a claim about *variance across wordings* into one about *ordering*. Third, a nonce-trained control and a training-time intervention probe the protocol from the training side (Appendix B.24).

Scope. Our testbed is the VLM2Vec lineage; MLLM-based universal embedding predates it (Jiang et al., 2024; Lin et al., 2024). Stronger current recipes, from hard-negative mining to reasoning-driven readout and late interaction (Appendix B.24), are not audited here; the protocol transfers to them with adapted scoring code (Section 7). VisIT-Bench (Bitton et al., 2023) varies instructions per instance, the design under which instruction following is directly measurable. MMEB-V2 (Meng et al., 2025) and ViDoRe (Faysse et al., 2025), like MMEB, fix one string per task (Appendix B.6).

3 The Audit: Six Conditions, Thirty-Six Tasks

Setup. All numbers use VLM2Vec-Qwen2VL-2B with LoRA weights and last-token pooling on the full 36-task MMEB (V1) image split. Our reproduction scores 59.59 against a reported 60.1 (59.15 on the 300-query grid used below), with a mean per-task deviation (1.58) inside the authors' own re-run noise floor (1.65) and 80.5% top-1 agreement with the shipped predictions (Appendix A). MMEB scores every task, whether classification, retrieval, VQA or grounding, as top-1 accuracy against the query's own candidate pool. Only pool size, and thus chance level, differs across tasks, so we report per-task numbers alongside every mean.

Protocol. Every task is evaluated under six conditions: *orig* (MMEB's own fixed instruction), three author-curated meaning-preserving *paraphrases* (para1–3), *swap* (another task's instruction verbatim, semantically wrong here), and *none* (instruction emptied). A condition's *gap* is its drop in mean top-1 accuracy below *orig*, in points: positive is a cost, negative a gain, and this sign is used throughout. The grid yields 217 scored cells (216 plus a train-wording cell for Section 4's natural experiment), each sampling the task's first 300 eval queries against its own official candidate pool (2 to 1,000 candidates; 1,000 on 26 of the 36 tasks). Rescoring the grid on all 1,000 eval queries per task moves no macro-mean gap by more than half a point (paraphrase 3.69, swap 9.54, deletion 7.70) and keeps the ordering (Appendix B.5).

Every condition changes the instruction on the **query side only**: candidate embeddings are encoded once from MMEB's unmodified candidate text and reused byte-identically, and the query image and content are held fixed, so only the query's instruction prefix varies within a task (Appendix B.1). This measures a query-side rewording against a fixed index, MMEB's own convention; systems that canonicalize queries, drop per-task instructions or reindex per query would face a different cost, and

Section 5 adds candidate-only and joint conditions. Each task's *swap* donor is one fixed task that shares the query-side modality but asks for a different task, with little lexical overlap (mean token Jaccard 0.26 with the recipient's instruction). Three further donors per task at graded lexical distances (mean Jaccard 0.52, 0.18 and 0.07) cost 13.16, 11.73 and 7.14 points and keep the ordering on 30 to 32 of 36 tasks, so lexical distance from the recipient's string does not predict the cost. A one-word meaning edit and a word shuffle cost about what a paraphrase does (2.99 and 3.66 against 3.32; Appendix B.2).

Table 1: Mean accuracy over all 36 tasks under the six conditions. Every value is a mean *over tasks*; the two paraphrase rows differ only in how a task's three paraphrases are aggregated, as their mean or their worst, *before* averaging over tasks. Higher gap means larger degradation.

Condition	Mean accuracy	Gap (degradation)
orig	59.15	—
paraphrase, task-mean of 3	55.82	3.32
paraphrase, task-worst of 3	53.40	5.75
swap (wrong-task instruction)	49.18	9.97
none (instruction deleted)	51.50	7.65

The average hides both failures and gains. A paraphrase costs 24.2 points on VOC2007 but gains 15.1 on ObjectNet and 12.2 on GQA. On these tasks the benchmark's own wording scores below an equivalent alternative. The task-worst paraphrase exposes a larger loss than the mean of three (Table 1). *The benchmark's chosen string is part of what determines the score.* The full per-task grid is in Appendix B.23.

The paraphrases mean the same thing. Three colleagues who are not authors rated all 252 instruction pairs blind to task and condition. Every rater scored 215 of the 216 paraphrase pairs as asking for the same task, and every rater rejected all 36 swap pairs (Fleiss' $\kappa = 0.82$; Appendix B.3). The one rejected paraphrase, a held-out DocVQA rewrite, is also that task's largest held-out drop. Restricting the audit to unanimously accepted pairs leaves the trained-on gaps unchanged (3.32 and 9.97) and moves the held-out gap from 3.33 to 3.27. Nor does the ordering depend on which paraphrases we wrote. Scoring each task under its most favourable paraphrase, the wrong-task instruction still costs more on 32 of 36 tasks, and dropping any one paraphrase slot from every task moves the mean paraphrase gap only between 3.02 and 3.84. The three paraphrase accuracies do spread within a task (mean range 4.57 points, exceeding the mean paraphrase gap itself on 21 of 36 tasks), and that spread is part of the finding: a benchmark's reported score moves by several points depending on which of three synonymous wordings its authors happened to choose.

Table 2: All audited checkpoints, 36 tasks $\times$ 300 queries (VLM2Vec-2B also on all 1,000 queries, Appendix B.5). Gaps are macro-mean points below *orig* (higher is worse). swap$-$para is the ordering margin with its cluster-resampled 95% interval (tasks, then queries; every gap's interval is in Table 11). Last column: tasks where swap costs more than paraphrase (sign-test p).

Checkpoint	orig	para	swap	none	swap$-$para [95% CI]	swap$>$para
VLM2Vec-Qwen2VL-2B	59.15	3.32	9.97	7.65	6.65 [3.92, 9.82]	31/36 (1.3×10^{-5})
VLM2Vec-Qwen2VL-7B	65.14	2.53	12.14	8.41	9.61 [5.35, 14.42]	28/36 (1.2×10^{-3})
Qwen3-VL-Embedding-2B	59.74	6.40	10.98	8.12	4.57 [2.48, 7.10]	29/36 (3.1×10^{-4})
Qwen3-VL-Embedding-8B	61.99	4.04	8.53	1.17	4.49 [1.75, 7.28]	25/36 (0.014)
GME-Qwen2VL-2B	51.55	1.41	2.07	0.49	0.67 [-3.74, 5.11]	18/36 (1.0)

The ordering extends beyond one checkpoint. A wrong instruction costs more than a paraphrase on 31 of 36 tasks for VLM2Vec-2B. Cluster-resampled intervals for this ordering clear zero on four of five checkpoints (Table 2); the table also reports the per-task counts. GME-Qwen2VL-2B is the exception, with smaller average costs and no clear ordering. Its small deletion gap partly reflects cancellation between task-level gains and losses, as detailed in Appendix B.4. *Template sensitivity depends on the training recipe, while the fixed-string protocol leaves it unmeasured in every case.*

Candidate-pool size changes the magnitude, not the main ordering. The paraphrase gap's magnitude concentrates in the three tasks with pools under 100 candidates: VOC2007 and N24News average 18.94 points and HatefulMemes, with a pool of 2, costs 6.22, so the headline 3.32 is partly a small-pool effect. The ordering is not. The 26 tasks with 1,000 candidates alone give a paraphrase gap of 2.74 and a swap gap of 8.40, a ratio of 3.07 against the pooled 3.00, and the small-pool stratum is where that ratio is lowest (1.30), so dropping the small pools sharpens the diagnostic rather than removing it. Adjusting for headroom above chance gives the same ordering. Appendix B.11 gives the strata and the reporting quantities.

Beyond MMEB-V1. MMEB-V2's visual-document split fixes one query instruction and one candidate template for all 27 of its datasets, so the convention holds at one string per meta-task. We ran the same six query-side conditions on every query of all 27 datasets, scored as MMEB-V2 scores them (nDCG@5), for three checkpoints (Table 3). The ordering replicates: a wrong-task instruction costs more than a paraphrase on 20 to 23 of 27 datasets for every checkpoint, with cluster-resampled intervals on the margin clear of zero. The magnitudes track exposure. Qwen3-VL-Embedding-2B pays 2.22 points for a paraphrase, positive on 26 of 27 datasets and concentrated in ViDoRe v2 (3.38 for a paraphrase and 10.08 for the wrong task over its 7 datasets). VLM2Vec-V2.0, released with this benchmark and trained with its string, pays 0.57 and loses more to deletion than to a wrong instruction, so it reads that an instruction is present rather than which one. VLM2Vec-Qwen2VL-2B, trained on MMEB-V1 only, has never seen the string: it gains 0.75 under a paraphrase while still paying 1.15 for a wrong one, a never-trained-wording control at benchmark scale. The same V2.0 weights, scored on MMEB-V1's 23 distinct per-task strings, lose 3.26 to a paraphrase and 11.85 to a wrong instruction (Appendix B.6). A string shared by 27 datasets can identify only the meta-task, so smaller gaps are what the exposure reading expects; we note the consistency without claiming the comparison as a test. Exploratory candidate-pool analyses and falsified method hypotheses sit in Appendices B.8, B.9 and B.18; no claim depends on them.

Table 3: MMEB-V2 visual-document audit: 27 datasets, all 12,917 queries, macro-mean nDCG@5 gaps below *orig* (higher is worse). Last column: datasets where the wrong-task instruction costs more than a paraphrase (sign-test p). Per-dataset values: Appendix B.6.

Checkpoint	Saw the string	orig	para	swap	none	swap>para
Qwen3-VL-Embedding-2B	reports MMEB-V2	73.25	2.22	4.82	3.23	23/27 (3.1×10^{-4})
VLM2Vec-V2.0	released with it	66.25	0.57	1.16	2.33	21/27 (5.9×10^{-3})
VLM2Vec-Qwen2VL-2B	no (V1 only)	41.28	-0.75	1.15	0.40	20/27 (0.02)

4 What the Rewording Cost Tracks: Exposure to the Exact String

A fixed instruction offers two signals: what the task asks and which wording the model has seen during training. We separate these signals by changing the trained wording, then test how much the benchmark rewards a cue with no linguistic meaning.

A lottery decides which wording becomes familiar. Each instruction group has two equivalent ways to state its task: MMEB's original wording, *orig*, and a fixed alternative, *rstr*. We randomly keep orig for some groups and replace it with rstr for others throughout training. Tasks sharing an instruction share the assignment; the lottery is stratified by classification and retrieval. VQA uses rstr by design. Two training seeds share this draw and otherwise follow the fixed-instruction recipe (`A_base`, §6). At evaluation, every task is tested with both wordings against the same candidate index. If training exposure drives preference, orig should win where it was kept and rstr where it replaced orig.

The preference reverses with the assignment. Original wording wins by 6.6 points on kept tasks (positive on 8 of 9) and loses by 1.3 on reassigned tasks (negative on 14 of 18; Figure 2). The sign reverses within classification and retrieval, so the pooled effect is not simply a contrast between task types. Over both types, the observed draw ranks first in the predicted direction among 700 allowed assignments (two-sided p=0.006); retrieval alone gives p=0.029. Classification alone permits just 10 draws, so it cannot reach significance: its first-ranked result has one-sided p=0.10 and two-sided p=0.20. Seed 1 alone gives p=0.081 over both types: the two seeds are two measurements of one

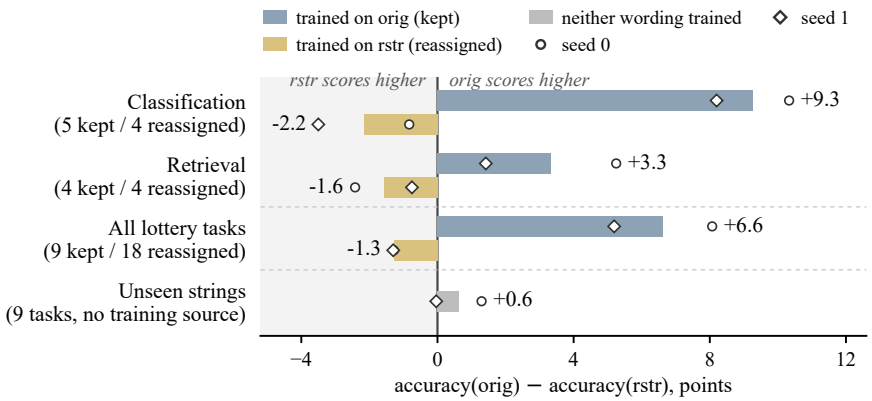


Figure 2: Wording preference reverses with assigned training exposure. Positive values favor MMEB's original wording (orig); negative values favor the fixed alternative (rstr). Bars average two training seeds sharing one assignment draw; markers show each seed, not confidence intervals. Classification and retrieval are the within-type comparisons; the pooled row adds VQA, which was reassigned by design. The nine tasks whose string has no training source show no preference, as predicted. All values come from the same experiment as Table 35.

Table 4: Two controls; higher gap means larger degradation. **(a)** Eight tasks, three seeds: a nonce-trained model nearly matches fixed-instruction training under its own cue. Gaps are below each model's score with MMEB's words. **(b)** VLM2Vec-2B on 30 image-query tasks, gaps below *orig*: blanking the image costs much more than rewriting the instruction. Full protocols: Appendices B.14 and B.7.

(a) What the protocol cannot detect

Arm Trained with	A_base fixed strings	D_tok nonce per string
Score, given its own cue	62.3	60.5
Score, given MMEB's words	62.3	54.2
Gap, paraphrase	7.31	0.33
Gap, held-out paraphrase	8.18	0.82
Gap, wrong task	15.6	10.8

(b) What the image carries

Condition	Gap
Paraphrase	3.8
No instruction	8.6
Wrong task	11.6
Blank image	**48.0**

randomization, not two experiments. Full assignment and per-task results are in Appendix B.29. *A wording gains an advantage when it becomes the trained wording.* Wording quality also matters: on kept tasks, the untrained rstr scores 3.5 points below the audit paraphrases.

Existing exposure patterns tell the same story. On the audited checkpoint, tasks whose exact evaluation string was absent from MMEB-train barely move under paraphrase, yet still lose accuracy under a wrong instruction. Grouping by exact string, rather than by the dataset's training split, separates these behaviors: mean absolute paraphrase gaps are 0.49 for unseen strings and 6.96 for seen strings (p=0.007), and no unseen task moves by more than 1.33 points. MMEB's own IND/OOD dataset split does not separate them: eight OOD datasets reuse a trained string and move 8.38 points on average, in both directions, against 0.50 for the eight that do not. The wrong-instruction cost remains on unseen tasks (5.67 points), so a never-trained string is inert to rewording but not to a wrong instruction. A complementary example is MSCOCO grounding, whose train and evaluation instructions differ. Restoring its training wording raises accuracy from 63.0 to 66.3, although a single pair of wordings cannot separate exposure from wording quality. These observations agree with the randomized test above; Appendix B.28 gives the exposure groups and statistical analysis.

A meaningless cue nearly matches meaningful instructions. We fine-tune `D_tok` with every natural-language query instruction replaced by a four-letter nonce, one per string. Candidate templates keep their words, and the base model was instruction-tuned beforehand. Given its trained nonce,

this model scores 60.5, versus 62.3 for fixed-wording training on the same eight tasks and three seeds (Table 4a). Its score remains competitive even though its fine-tuning replaced the instructions' words. Given MMEB's real words instead, it scores 54.2, six points below its own cue: for this model the meaningful instruction behaves like an unfamiliar paraphrase, yet a wrong instruction still costs it 10.75 points and a one-word meaning change 6.50, while deleting the instruction costs it nothing (-0.07). Two further arms sharpen the picture (Appendix B.14). `E_fixtok`, trained with the nonce prefixed to the fixed instruction, uses both cues: removing the nonce costs 1.4 points and keeping it trims the paraphrase gap by little more than a point. `F_augtok`, paraphrase sampling with a constant nonce, tests whether a fixed identifier absorbs task identity once the words vary. It does not: its held-out gap falls from `B_aug`'s 3.55 to 2.50, not to zero, and with words alone it pays 0.94 for a rewording, 8.10 for a one-word meaning change and 15.93 for the wrong task, the ordering an instruction-follower should show. A wrong nonce costs `D_tok` 7.46 points, about half of what a wrong instruction costs `A_base` (15.64). Models trained with words use the words; a constant identifier is used when present but does not replace them. The control demonstrates what fixed-wording scoring can reward; it does not identify the internal mechanism of another model.

The image still carries the task content. Blanking the query image costs 48.0 points, compared with 3.8 for a paraphrase on the same image-query tasks (Table 4b). On the two tasks that reuse an image across questions (GQA, InfographicsVQA), a query's embedding prefers its own answer over another question's answer to the same image in 84.5% and 89.4% of pairs, so the model is steered by what is asked. *Reliance on a familiar instruction can coexist with strong use of image and query content.* The image-side and repeated-image controls are detailed in Appendices B.7 and B.26.

From exposure to embedding geometry. Template-only edits move embeddings while content stays fixed. Task identity accounts for 32% of the variance of both query and candidate embeddings. Removing a single period from a candidate template ("image." to "image") moves the candidate embedding to cosine 0.74 with its original, against a noise floor of 0.9994, and costs MSCOCO text-to-image 3.5 points; the same edit on an already-canonical template (VisDial) leaves cosine at 0.9994 and accuracy unchanged. On SUN397, a paraphrase and a wrong instruction displace the query embedding by a similar amount (cosine 0.610 and 0.620 to the original). A learned cue can therefore affect ranking through the geometry of the representation. A linear probe for the instructed target type, trained on some strings and tested on others, reaches 78.3% at the final layer against 59.3% with the instruction deleted, and no internal layer decodes type better (Appendix B.30); the probe cannot separate a template identifier from a representation of type, and we do not claim that it does. A rival explanation is misalignment between the rewritten query and candidates encoded under the original template; the next section tests it by rewriting both sides. Additional geometric probes and training-free repairs are in Appendices B.27, B.30, and B.17.

5 Which Side Carries the Cost: Candidate and Joint Rewrites

The audit rewrites the query instruction while the candidate index stays encoded under MMEB's original template. The cost could therefore reflect misalignment: the rewritten query no longer matches candidates encoded under the old template, even if the model reads the instruction correctly (*template-alignment*). Alternatively, each rewrite damages its own side's embedding. Nothing forces a shared instruction to cancel: writing $z_q = f(q, I_T)$ and $z_c = f(c, I_T)$, I_T enters $\cos(z_q, z_c)$ through the inner product and both norms, so rewriting it on both sides need not restore the ranking. The two accounts make opposite predictions about one experiment.

Design. We add *cand-parak* (rewrite the candidate template, leave the query original) and *joint-parak* (rewrite both sides, query paraphrase k paired with candidate paraphrase k), reusing the candidate-side paraphrases of §6. The conditions apply to the 10 of 36 tasks whose candidate text is a constant template alone. Four more prefix a template to per-candidate text and were not run; on the other 22 the candidate is bare content and the condition is undefined. Each cell re-encodes the full candidate pool and scores 300 queries (Appendix B.21). Template-alignment predicts the *joint* gap collapses toward zero; independent per-side damage predicts roughly the *sum* of the single-side gaps. The predictions differ by the full size of the effect.

Changing both sides leaves most of the cost. The joint gap is 5.46 points, with a task-resampled 95% interval of $[1.68, 10.02]$. It stays well above the zero predicted by full template-alignment

Table 5: Paraphrase gaps (points below *orig*; higher is worse) by rewritten side, over the 23 cells with paired per-query records (10 tasks $\times$ available candidate paraphrases). $\Delta_{\mathrm{Q}} + \Delta_{\mathrm{C}}$ is the independent per-side damage prediction for the joint condition; template-alignment predicts $\Delta_{\mathrm{J}} \approx 0$; the last column is $\Delta_{\mathrm{J}} - (\Delta_{\mathrm{Q}} + \Delta_{\mathrm{C}})$. Per-task rows and cluster-bootstrap intervals: Appendix B.21 (Table 30).

	n cells	Δ_{Q} (query)	Δ_{C} (cand)	Δ_{J} (joint)	$\Delta_{\mathrm{Q}} + \Delta_{\mathrm{C}}$	inter.
all 10 tasks	23	**3.33**	**3.16**	**5.46**	**6.49**	$\mathbf{-1.03}$

recovery, and is closer to the additive prediction of 6.49. *Matching the rewrite on both sides does not restore the original ranking.*

Across the paired cells, the joint gap is about 84% of the sum of the single-side gaps ($\alpha = 0.84$, the through-origin slope of joint on summed gaps; task-and-query bootstrap 95% CI $[0.62, 0.91]$). The slope weights large-gap cells such as CIRR most. The unweighted mean interaction is -1.03 points, and its interval under the same resampling crosses zero; it is negative in 16 of the 23 cells, zero in 2 and positive in 5, and its per-cell interval excludes zero in only 2. This supports a rejection of full recovery while leaving room for partial alignment effects. On half of the tasks, a candidate rewrite costs more than a query rewrite, so the query-only audit can understate total template dependence. The asymmetry runs both ways: CIRR loses 20.6 points to a query rewrite but 2.6 to a candidate rewrite, while MSCOCO text-to-image loses 1.9 and 6.6. Per-task effects and bootstrap details are in Appendix B.21.

Scope. This test uses one checkpoint and one seed, and covers 23 of the 30 possible cells; the 4 tasks that prefix a template to per-candidate text were not run.

6 A Training-Time Intervention

If the cost comes from the string's constancy in training, the minimal intervention removes that constancy and leaves every other training choice as it is. It samples a fresh meaning-preserving paraphrase of the instruction on every query and candidate at every training step (`instr_aug`). Augmentation of this kind is standard; its value here is that it supplies the counterfactual the evaluation side cannot run. Ngweta et al. (2025) mix prompt *formats* for text-only LLMs, a complementary intervention. A variant that adds a consistency loss between paraphrase views lowers the held-out gap slightly further (3.14 against 3.55) but costs 1.5 points of fixed-wording accuracy, so the main intervention is paraphrase sampling alone; the variant is reported as an ablation in Appendix B.13.

Objective. The base objective is InfoNCE over in-batch negatives at temperature $\tau = 0.02$, on last-token-pooled, L_2-normalized query and candidate embeddings. With `instr_aug`, every item independently draws one of four instruction variants for its query and, separately, one for its positive candidate (which has its own four-variant template set), so a query and its positive are not forced to share a wording; the three non-original variants are the same author-curated paraphrases used in the audit. Appendices B.25 and B.16 give the full objective and hyperparameters.

Setup and held-out evaluation. We compare fixed-instruction training (`A_base`) with paraphrase sampling (`B_aug`), using the same LoRA recipe, 2000 training steps, and three seeds on an eight-task subset. MMEB-train and MMEB-eval are disjoint. Each task has a separate set of three paraphrases never sampled in these arms' training or used for model selection. This tests whether augmentation teaches tolerance beyond the specific rewrites it sees. The all-task held-out evaluation uses one seed; full setup and seed-level results are in Appendices B.16 and B.13.

The benefit transfers to new wording. On held-out paraphrases, augmentation lowers the mean gap from 8.18 to 3.55 on the eight-task subset, with a reduction at every seed (9.43, 8.35 and 6.76 to 3.32, 3.97 and 3.35). The mean wrong-instruction gap remains near the fixed-training level (Table 6). The 36-task evaluation agrees: the held-out gap falls from 3.33 to 1.56. The effect appears early: at 500 steps and one seed, sampling already lowers the paraphrase gap from 3.85 to 1.66, but there it also lowers the wrong-task gap from 11.92 to 6.62, a sensitivity the protocol should keep; at 2000 steps that gap stays near the fixed level, so the longer budget is the one we report. *Varying wording during training reduces the penalty for saying the same thing differently, while retaining the mean penalty for asking the wrong task.*

Table 6: Paraphrase sampling at 2000 steps, eight tasks, mean of three seeds. Gaps are accuracy points below orig: higher means larger degradation. Held-out paraphrases are excluded from training and model selection. The consistency-loss ablation and per-seed results are in Appendix B.13.

Arm	Held-out gap	Trained-on gap	Wrong-task gap	No-instr. gap	Orig
Fixed	8.18	7.31	15.6	9.28	62.3
Sampled	3.55	0.24	15.2	6.72	61.8

Coverage improves; invariance remains incomplete. The trained-on gap shrinks much more than the held-out gap, showing that augmentation learns useful wording coverage without making all equivalent instructions interchangeable. A one-seed, 500-step ablation that augments only one side finds that query-side sampling accounts for nearly all of the paraphrase-gap closure (3.85 to 1.36, against 1.66 when both sides are sampled; Appendix B.15). Trained-on paraphrases nearly vanish (0.24), so a benchmark that reused an augmentation's own paraphrases would overstate invariance. A residual held-out cost therefore remains an evaluation target. For benchmark design, the implication is practical: use unseen paraphrases to test wording tolerance, and retain wrong-task instructions as a separate check that the model still distinguishes what is being asked.

7 Limitations

The audit and the nonce control are direct measurements; the lottery shows that training on a wording privileges it, not that the model then uses it as a task identifier, and that reading remains entangled with instruction semantics and embedding alignment. The lottery covers 13 randomized string groups, one recipe and one draw shared by two seeds, and is significant over both task types and in retrieval but not in classification alone, which allows only 10 draws. Three LLM-drafted, author-screened held-out strings per task cover a small part of the wording space, and the raters agreed on meaning but not on well-formedness ($\kappa \approx 0$; on the 26 unanimously accepted tasks the ordering still holds, 2.03 against 6.60). Wrong-task costs depend on the donor, so swap against paraphrase is an ordering, not a calibrated amount of understanding. The exposure, side-rewrite and training interventions use one checkpoint and three seeds on eight tasks (one seed on 36); late-interaction, reasoning-driven and hard-negative-mined recipes remain outside the audit, and GME's smaller costs already show recipe dependence. Augmentation also widens wording coverage, so its benefit combines that coverage with reduced fixedness; the token arms separate the two only partly. Task-resampled audit intervals and training-seed ranges describe different sources of variation. Subsets were chosen after the audit. MMEB-V1 scores top-1 accuracy and the visual-document split nDCG@5, so across benchmarks we compare orderings, not magnitudes. Systems that canonicalize instructions or rebuild their index under a new query convention require a corresponding audit (Appendix B.20).

8 Conclusion

A fixed instruction is both a description of a task and a familiar string, and MMEB's scoring leaves these roles entangled. Varying the instruction reveals a rewording cost on all five MMEB-V1 checkpoints, and a wrong instruction costs reliably more on four of them; these are direct measurements of a blind spot in the protocol. Assigning training wording by lottery reverses wording preference, and a meaningless task cue nearly matches fixed-instruction training. Learned dependence on the exact string, alongside instruction meaning, is the reading most consistent with these results. Paraphrase sampling more than halves the held-out cost at almost no cost in fixed-wording accuracy (62.3 against 61.8), while leaving a residual that fixed-wording accuracy cannot show; the held-out gap, not the trained-on gap, is the quantity to track, since the latter falls to 0.24 while the former stays at 3.55. Instruction-conditioned benchmarks should report paraphrase and wrong-instruction gaps beside their accuracy scores; the released audit needs only the per-task instruction field and the benchmark's own candidate index, and each added condition is one more pass over the queries against an index that is encoded once. The audit scripts and every instruction string we evaluated are released with the paper.

REPRODUCIBILITY STATEMENT

Every number in Sections 3–6 is produced by the released 6-condition pipeline (`perturb_eval.py`, `perturb_report.py`, `instructions.py`) described in Appendix A, which also documents the reproduction of the shipped VLM2Vec-Qwen2VL-2B checkpoint (59.59 vs. reported 60.1) and the `image_resolution=high` setting that the reproduction depends on. All audited checkpoints (VLM2Vec-Qwen2VL-2B/7B, VLM2Vec-V2.0, GME-Qwen2VL-2B, Qwen3-VL-Embedding-2B/8B), the MMEB-V1 evaluation and training splits and the MMEB-V2 visual-document split are publicly available. Training details for the `instr_aug` and `inv_weight` arms — LoRA configuration, step budgets, batch sizes, learning rate, the augmentation sampling rule, and the consistency-loss weight λ — are given in Appendix B.16; per-seed results are in Appendix B.13, paired bootstrap intervals in Appendix B.4, the full-query replication of the VLM2Vec-2B grid in Appendix B.5, the image-side conditions in Appendix B.7, and the full per-task grid in Appendix B.23. The anonymized supplementary material contains the audit and training scripts, the complete instruction set (original, three paraphrases, and all swap donors per task), and the JSON summaries underlying the reported tables and the numerical results in Figures 1 and 2. Figure 1 embeds square-cropped example images from the MMEB VOC2007 evaluation split; its displayed values were checked against the result tables.

ETHICS STATEMENT

This work audits the evaluation protocol of a public benchmark using publicly released model checkpoints and data; it involves no personal data and no new data collection beyond author-curated instruction paraphrases and a blinded rating of those paraphrases by three colleagues who are not authors (instruction strings only; ratings reported in aggregate). Our findings concern a protocol convention, not the conduct of the benchmark's or the models' authors, and we state the scope of the claim explicitly in Section 7. The released audit protocol is intended to make instruction-following claims easier to verify, and we see no direct path to harmful use.

AI USE STATEMENT

In this work, we used generative AI tools — a large-language-model coding agent operated by the authors — for writing assistance and drafting of the manuscript, which the authors then revised; for generation and refactoring of the evaluation, analysis, and training code; for orchestrating and monitoring experiment runs; for retrieval and discovery of related work; and for generating and editing the schematic artwork of Figure 1 with an image-generation tool. The photographs in that figure are unaltered crops of MMEB images, and its displayed values were checked against the result tables. The benchmark data (images, queries, candidates and MMEB's original instructions) are the public releases and were not generated or altered by us. The instruction rewrites we evaluate — the trained-on and held-out query paraphrases, the candidate-template paraphrases, the one-word *minedit* edits and the hand-picked swap-donor assignment — were drafted with the same LLM assistance and screened by the authors before use (for meaning preservation, or for *minedit* a changed task); we describe them as author-curated rather than hand-written for this reason. The *shuffle* and nonce conditions are generated by seeded code, and swap strings are other tasks' original MMEB instructions. Apart from these rewrites, we have not used generative AI tools to generate evaluation data, to produce or alter any reported number, or to prove mathematical claims, and the remaining disclosure categories are not applicable to this work. We have reviewed all AI-assisted work: every reported number was produced by the released code from disk-backed artifacts, LLM-assisted code was checked against the shipped VLM2Vec predictions (Appendix A), and every citation was verified against the cited source before inclusion. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. Task-aware retrieval with instructions. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.

Matteo Attimonelli, Alessandro De Bellis, Aryo Pradipta Gema, Rohit Saxena, Monica Sekoyan, Wai-Chung Kwan, Claudio Pomo, Alessandro Suglia, Dietmar Jannach, Tommaso Di Noia, and Pasquale Minervini. Do

composed image retrieval benchmarks require multimodal composition? *arXiv preprint arXiv:2605.14787*, 2026.

Yonatan Bitton, Hritik Bansal, Jack Hessel, Rulin Shao, Wanrong Zhu, Anas Awadalla, Josh Gardner, Rohan Taori, and Ludwig Schmidt. VisIT-Bench: A benchmark for vision-language instruction following inspired by real-world use. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

Libiao Chen et al. UMER: Unifying embedding and ranking via pair-aware discriminative reasoning for universal multimodal retrieval. *arXiv preprint arXiv:2608.18504*, 2026.

Jaekeol Choi. When does supervised fine-tuning reduce instruction sensitivity? *arXiv preprint arXiv:2608.26661*, 2026.

Xuanming Cui, Jianpeng Cheng, Hong-you Chen, Satya Narayan Shukla, Abhijeet Awasthi, Xichen Pan, Chaitanya Ahuja, Shlok Kumar Mishra, Yonghuan Yang, Jun Xiao, Qi Guo, and Ser-Nam Lim. Think then embed: Generative context improves multimodal embedding. *arXiv preprint arXiv:2510.05014*, 2025.

Xuanming Cui, Shlok Kumar Mishra, Wentao Bao, Aashu Singh, Zihao Wang, Xiangjun Fan, Jun Xiao, Ser-Nam Lim, and Jianpeng Cheng. MoEMB: Scaling universal multimodal embeddings with efficient mixture-of-experts models. *arXiv preprint arXiv:2609.08663*, 2026.

Elliot L. Epstein, Kaisheng Yao, Jing Li, Xinyi Bai, and Hamid Palangi. MMMT-IF: A challenging multimodal multi-turn instruction following benchmark. *arXiv preprint arXiv:2409.18216*, 2024.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. ColPali: Efficient document retrieval with vision language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Manuel Frank and Haithem Afli. PTEB: Towards robust text embedding evaluation via stochastic paraphrasing at evaluation time with LLMs. *arXiv preprint arXiv:2510.06730*, 2025.

Geonmo Gu, Byeongho Heo, Jaemyung Yu, Jaehui Hwang, Taekyung Kim, Sangmin Lee, HeeJae Jun, Yoohoon Kang, Sangdoo Yun, and Dongyoon Han. MuCo: Multi-turn contrastive learning for multimodal embedding model. *arXiv preprint arXiv:2602.06393*, 2026.

Tiancheng Gu, Kaicheng Yang, Ziyong Feng, Xingjun Wang, Yanzhao Zhang, Dingkun Long, Yingda Chen, Weidong Cai, and Jiankang Deng. Breaking the modality barrier: Universal embedding learning with multimodal LLMs. *arXiv preprint arXiv:2504.17432*, 2025a.

Tiancheng Gu, Kaicheng Yang, Kaichen Zhang, Xiang An, Ziyong Feng, Yueyi Zhang, Weidong Cai, Jiankang Deng, and Lidong Bing. UniME-V2: MLLM-as-a-Judge for universal multimodal embedding learning. *arXiv preprint arXiv:2510.13515*, 2025b.

Jinjin Guo, Yexin Li, Zhichao Huang, Jun Fang, Zhiyuan Liu, Chao Liu, Pengzhang Liu, and Qixia Jiang. Spectral disentanglement and enhancement: A dual-domain contrastive framework for representation learning. In *Proceedings of the ACM Web Conference (WWW)*, 2026.

Chenwei He, Xiangzhao Hao, Tianyu Yang, Yuxiang Ma, Yuheng Jia, Lingxiang Wu, Chaoyang Zhao, Haiyun Guo, and Jinqiao Wang. PLUME: Latent reasoning based universal multimodal embedding. *arXiv preprint arXiv:2604.02073*, 2026.

Haohang Huang, Xuan Lu, Mingyi Su, Xuan Zhang, Ziyan Jiang, Ping Nie, Kai Zou, Tomas Pfister, Wenhu Chen, Wei Zhang, Xiaoyu Shen, and Rui Meng. MMEB-V3: Measuring the performance gaps of omni-modality embedding models. *arXiv preprint arXiv:2604.23321*, 2026.

Mohamed Insaf Ismithdeen, Muhammad Uzair Khattak, and Salman Khan. Promptception: How sensitive are large multimodal models to prompts? In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

Ting Jiang, Minghui Song, Zihan Zhang, Haizhen Huang, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and Fuzhen Zhuang. E5-V: Universal embeddings with multimodal large language models. *arXiv preprint arXiv:2407.12580*, 2024.

Ziyan Jiang, Rui Meng, Xinyi Yang, Semih Yavuz, Yingbo Zhou, and Wenhu Chen. VLM2Vec: Training vision-language models for massive multimodal embedding tasks. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.05160.

Song Jin, Zhongtao Jiang, Chenglei Shen, Huanxuan Liao, Haozhe Chi, Zhiwei Wang, Kun Xu, and Yong Liu. Allocate before you embed: Adaptive visual input allocation for video embeddings. *arXiv preprint arXiv:2609.01778*, 2026.

Sungyeon Kim, Xinliang Zhu, Xiaofan Lin, Muhammet Bastan, Douglas Gray, and Suha Kwak. GENIUS: A generative framework for universal multimodal search. *arXiv preprint arXiv:2503.19868*, 2025.

Yevhen Kostiuk and Kenneth Enevoldsen. One prompt is not enough: Instruction sensitivity undermines embedding model evaluation. *arXiv preprint arXiv:2605.22544*, 2026. Version 2, 13 August 2026.

Zhibin Lan, Liqiang Niu, Fandong Meng, Jie Zhou, and Jinsong Su. UME-R1: Exploring reasoning-driven generative multimodal embeddings. *arXiv preprint arXiv:2511.00405*, 2025.

Qi Li, Yanzhe Zhao, Yongxin Zhou, Yameng Wang, Yandong Yang, Yuanjia Zhou, and Jinxiang Liu. Magic-MM-Embedding: Towards visual-token-efficient universal multimodal embedding with MLLMs. *arXiv preprint arXiv:2602.05275*, 2026a.

Weien Li, Rui Song, Zeyu Li, Haochen Liu, Gonghao Zhang, Difan Jiao, Zhenwei Tang, Bowei He, Haolun Wu, Xue Liu, and Ye Yuan. MINER: Mining multimodal internal representation for efficient retrieval. *arXiv preprint arXiv:2605.06460*, 2026b.

Xiaojie Li, Chu Li, Shi-Zhe Chen, and Xi Chen. U-MARVEL: Unveiling key factors for universal multimodal retrieval via embedding learning with MLLMs. In *International Conference on Learning Representations (ICLR)*, 2026c.

Yiwei Li, Jiayi Shi, Shaoxiong Feng, Peiwen Yuan, Xinglin Wang, Boyuan Pan, Heda Wang, Yao Hu, and Kan Li. Instruction embedding: Latent representations of instructions towards task identification. *arXiv preprint arXiv:2409.19680*, 2024.

Sheng-Chieh Lin, Chankyu Lee, Mohammad Shoeybi, Jimmy Lin, Bryan Catanzaro, and Wei Ping. MM-Embed: Universal multimodal retrieval with multimodal LLMs. *arXiv preprint arXiv:2411.02571*, 2024.

Rui Meng, Ziyan Jiang, Ye Liu, Mingyi Su, Xinyi Yang, Yuepeng Fu, Can Qin, Zeyuan Chen, Ran Xu, Caiming Xiong, Yingbo Zhou, Wenhu Chen, and Semih Yavuz. VLM2Vec-V2: Advancing multimodal embedding for videos, images, and visual documents. *arXiv preprint arXiv:2507.04590*, 2025.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. State of what art? A call for multi-prompt LLM evaluation. *Transactions of the Association for Computational Linguistics*, 2024.

Elio Musacchio, Giovanni Semeraro, and Goran Glavaš. Mind your special tokens! On the importance of dedicated sequence-end tokens in vision-language embedding models. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL), Short Papers*, 2026.

Juan Miguel Navarro Carranza. LLMs show surface-form brittleness under paraphrase stress tests. *arXiv preprint arXiv:2510.08616*, 2025.

Lilian Ngweta, Kiran Kate, and Jason Tsay. Towards LLMs robustness to changes in prompt format styles. *arXiv preprint arXiv:2504.06969*, 2025.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 3845–3864, 2023.

Jiajun Qin, Yuan Pu, Zhuolun He, Seunggeun Kim, David Z. Pan, and Bei Yu. UniMoCo: Unified modality completion for robust multi-modal embeddings. *arXiv preprint arXiv:2505.11815*, 2025.

Amirhossein Razavi, Mina Soltangheis, Negar Arabzadeh, Sara Salamat, Morteza Zihayat, and Ebrahim Bagheri. Benchmarking prompt sensitivity in large language models. *arXiv preprint arXiv:2502.06065*, 2025.

Angelika Romanou, Mark Ibrahim, and Candace Ross. Brittlebench: Quantifying LLM robustness via prompt sensitivity. *arXiv preprint arXiv:2603.13285*, 2026.

Emanuel Sánchez Aimar, Gulnaz Zhambulova, Fahad Shahbaz Khan, Yonghao Xu, and Michael Felsberg. VLM2GeoVec: Toward universal multimodal embeddings for remote sensing. *arXiv preprint arXiv:2512.11490*, 2025.

Daiki Shiono, Shumpei Miyawaki, Ryota Tanaka, and Jun Suzuki. Instruction-following evaluation of large vision-language models. *arXiv preprint arXiv:2512.23572*, 2025.

Rui Song, Yingji Li, and Lida Shi. Shortcut learning in in-context learning: A survey. *arXiv preprint arXiv:2411.02018*, 2024.

Siyu Sun, Jing Ren, Zhaohe Liao, Dongxiao Mao, Xiangyuan Ren, Yiyi Zhang, Haohua Zhao, Weixiong Lin, Jiang Shaohua, Liqing Zhang, and Yuchao Zheng. Bottleneck tokens for unified multimodal retrieval. *arXiv preprint arXiv:2604.11095*, 2026.

Abdul Basit Tonmoy, Kazi Fardinul Hoque, Md. Shahrier Islam Arham, and Arman Luthra. Fusion embedding: A unified embedding space for text, image, video, and audio. *arXiv preprint arXiv:2607.18666*, 2026.

Shijie Wang, Xiangzhao Hao, Yueti Li, Guangyu Cao, Xinyu Tang, and Haiyun Guo. ReLoop-UME: Recurrent depth with learnable retrieval registers for universal multimodal embedding. *arXiv preprint arXiv:2607.28751*, 2026a.

Yuchi Wang, Haiyang Yu, Weikang Bian, Jiefeng Long, Xiao Liang, Chao Feng, and Hongsheng Li. MMEmb-R1: Reasoning-enhanced multimodal embedding with pair-aware selection and adaptive control. *arXiv preprint arXiv:2604.06156*, 2026b.

Zhicheng Wang, Chen Ju, Xu Chen, Shuai Xiao, Jinsong Lan, Xiaoyong Zhu, Ying Chen, and Zhiguo Cao. Explore more, learn better: Parallel MLLM embeddings under mutual information minimization. *arXiv preprint arXiv:2511.01588*, 2025.

Peixi Wu, Biao Yang, Feipeng Ma, Bosong Chai, Bo Lin, Wei Yuan, Fan Yang, Tingting Gao, Hebei Li, and Xiaoyan Sun. LaME: Learning to think in latent space for multimodal embedding via information bottleneck. *arXiv preprint arXiv:2606.13061*, 2026a.

Yebo Wu, Feng Liu, Ziwei Xie, Zhiyuan Liu, Changwang Zhang, Jun Wang, and Li Li. TSEmbed: Unlocking task scaling in universal multimodal embeddings. *arXiv preprint arXiv:2603.04772*, 2026b.

Zilin Xiao et al. MetaEmbed: Scaling multimodal retrieval at test-time with flexible late interaction. *arXiv preprint arXiv:2509.18095*, 2025.

Zhiyang Xu, Ying Shen, and Lifu Huang. MultiInstruct: Improving multi-modal zero-shot learning via instruction tuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

Youze Xue, Dian Li, and Gang Liu. Improve multi-modal embedding learning via explicit hard negative gradient amplifying. *arXiv preprint arXiv:2506.02020*, 2025.

Xiaocui Yang, Wenfang Wu, Shi Feng, Ming Wang, Daling Wang, Yang Li, Qi Sun, Yifei Zhang, Xiaoming Fu, and Soujanya Poria. MM-InstructEval: Zero-shot evaluation of (multimodal) large language models on multimodal reasoning tasks. *arXiv preprint arXiv:2405.07229*, 2024.

Xu Yang, Chen Liu, and Ying Wei. Data selection matters: Towards robust instruction tuning of large multimodal models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Junjie Zhou, Ke Mei, Lei Li, Tianyi Wang, Fengyun Rao, and Jing Lyu. WeMM-Embedding: WeChat multimodal embedding technical report. *arXiv preprint arXiv:2608.24053*, 2026.

A Reproduction Details and the MMEB-Instruct Protocol

A.1 Reproduction validity

Full 36-task evaluation of the shipped VLM2Vec-Qwen2VL-2B checkpoint scores 59.59 overall against the authors' reported 60.1 (-0.5), with mean absolute per-task difference 1.58 against the authors' own re-run-vs-shipped noise floor of 1.65, and 80.5% top-1 prediction agreement with the shipped predictions file. The 59.59 is the reproduction run on every query (`runs/vlm2vec2b_high`); the audit pipeline, run separately on the same 1,000 queries per task, gives 59.66 (Appendix B.5), and the 300-query grid gives 59.15, the *orig* of every audit table. Achieving this required setting `image_resolution=high`, which force-resizes every image to a 1344×1344 square before the processor's own smart-resize step; without it, mean absolute per-task difference rises to 10.3 and grounding tasks drop 10–26 points. Guo et al. (2026) report the 336 and 1344 settings, and Qin et al. (2025) set VLM2Vec's 1344 variant aside as a fair-comparison confound, but neither notes that the setting discards aspect ratio on a backbone (Qwen2-VL) with native dynamic-resolution support that does not require a fixed square input. We report this as a supporting reproducibility finding, appended here rather than in the main text, because it does not bear on the paper's central instruction-sensitivity claim.

A.2 THE MMEB-INSTRUCT PROTOCOL (C2)

We release the 6-condition audit as `code/perturb_eval.py` and `code/perturb_report.py`: a drop-in for MMEB-style benchmarks that scores a checkpoint under orig, three paraphrases, swap, and none, reporting orig accuracy, mean-paraphrase accuracy, and worst-paraphrase accuracy per task. All six conditions rewrite the query-side instruction only, against a candidate index encoded once from the benchmark's unmodified candidate text (Table 7). All paths and the checkpoint identifier are command-line flags rather than hardcoded values. `instructions.py` supplies coverage for all 36 MMEB-V1 tasks. Every number in Sections 3–6 was produced by this pipeline. The paraphrases are held out with respect to any checkpoint that did not train on them, which covers every released checkpoint we audit in Section 3; they are *not* held out with respect to our own `instr_aug` arms in Section 6, which sample from the same three strings during training, and a user of this protocol who also trains with paraphrase augmentation should reserve distinct wordings for evaluation. The intended use is to report mean- and worst-paraphrase accuracy and the wrong-instruction gap beside *orig* accuracy, so that a reader can see how much of a fixed-wording score depends on the wording. The released scripts cover MMEB-V1's 36 tasks and, through `visdoc_perturb_audit.py`, MMEB-V2's visual-document split (Appendix B.6); another benchmark needs its own paraphrase set and loader. Sensitivity to the wording does not by itself separate memorization from instruction-following (Section 7).

B AUDIT AND METHOD DETAILS

This appendix holds the protocol, hyperparameter, and analysis specifications referenced from Sections 3–6. Most of it is the detail a reader would need to reimplement or contest the main-text numbers; supporting analyses the main text only summarizes (e.g. the graded swap controls) are included, and exploratory ones are labelled.

B.1 CONDITION × SIDE MATRIX

All six audit conditions rewrite the instruction on the **query side only** (`code/perturb_eval.py`). Candidate embeddings are encoded once from MMEB's own unmodified candidate text, cached, and reused byte-identically across all six conditions; the query image and the query's own content (the question, the class-label prompt, the caption) are likewise held fixed, so the only thing that varies within a task is the instruction prefix of the query string.

Table 7: Condition × side. Every condition perturbs the query encoder call only; the candidate index is byte-identical across all six.

Condition	Query-side instruction text	Candidate-side text
orig	MMEB's fixed string I_T	MMEB original (unmodified)
para1–3	author-curated paraphrase of I_T	MMEB original (unmodified)
swap	donor task's string $I_{D(T)}$, verbatim	MMEB original (unmodified)
none	instruction removed, content kept	MMEB original (unmodified)

This is a deliberate choice — it makes the measured change attributable to the instruction text alone — but it is also a scope restriction. What we report is the cost of a query-side instruction change under a fixed candidate index, which is the setting a deployed retriever faces when a user rewords a query against a pre-built index. The six conditions of the main grid are all query-side; the candidate-only and jointly-perturbed variants are a separate grid, run on the 10 tasks whose candidates carry a rewritable template and reported in Section 5 and Appendix B.21, and no Section 3 number covers them. Note also that MMEB's candidate text is not the query instruction: for most tasks it is the raw label or answer string, and for the grounding and image-target tasks it is a separate per-task constant template (Table 33).

B.2 SWAP DONOR SELECTION, AND ROBUSTNESS TO IT

Each task is assigned one primary donor task, fixed for the whole audit and released as a lookup table in `instructions.py`. The donor always shares the query-side modality with the recipient, so *swap* never also changes what kind of input the model is asked to embed, and is chosen by inspection to ask for a different task, so the substitution is semantically wrong rather than a near-paraphrase in disguise; it can still share tokens with the recipient's instruction (mean Jaccard overlap 0.26, Table 8).

That single hand-picked donor cannot distinguish "wrong instructions are expensive" from "this donor is expensive", so we added three further donors per task, chosen with no hand-picking at all. For each task we rank every other task whose instruction body is not byte-identical to this one's by Jaccard overlap of lowercased alphanumeric tokens, and take the highest (*near*), median-ranked (*mid*) and lowest (*far*); ties break on task name, so the table is a deterministic function of the released instruction set (`instructions.SWAP_STRATA`). Byte-identical donors are excluded because MMEB reuses one string across whole task groups — all 10 VQA tasks share one instruction, as do ImageNet-1K/A/R, VOC2007/ObjectNet and SUN397/Place365 — and substituting an identical string is not a perturbation at all. The three strata separate cleanly by construction (mean overlap 0.52 / 0.18 / 0.07, against 0.26 for the hand-picked donor). All 36 tasks were re-run under all three, 300 queries each, query side only, against the same cached candidate embeddings as every other condition.

Table 8: Swap gap under four donors per task, all 36 tasks. *Overlap* is the mean Jaccard token overlap between donor and original instruction; *ratio* is against the 3.32 paraphrase gap; *cos* is mean cosine between the perturbed and original query embedding (paraphrase: 0.78).

Donor	Overlap	Mean gap	Ratio	Tasks hurt	cos
near (highest overlap)	0.522	13.16	3.96×	31/36	0.637
mid (median overlap)	0.180	11.73	3.53×	32/36	0.599
far (lowest overlap)	0.074	7.14	2.15×	30/36	0.650
hand-picked (main text)	0.255	9.97	3.00×	31/36	0.667

Three readings, in decreasing order of how well they are supported.

The ordering the paper rests on is not an artifact of donor choice. Every stratum's mean gap exceeds the 3.32 paraphrase gap, by $2.15\times$ to $3.96\times$, and the hand-picked donor's $3.0\times$ is middling rather than favourable — two of the three unpicked strata are worse for the model than the donor we chose. The count of tasks hurt is also stable (30–32 of 36).

Lexical distance does not predict the cost, and closer is not safer. Across all 144 task×donor cells the rank correlation between overlap and gap is only $+0.135$, and the apparent monotone ordering in the table (13.16 / 11.73 / 7.14) does not survive removing three tasks: excluding ImageNet-1K/A/R gives 7.98 / 11.08 / 6.84, i.e. *near* drops below *mid*. Those three are a structural artifact worth naming rather than hiding. Their nearest donor is N24News, whose instruction ends in a connector expecting a caption ("…for domain classification: "), while an ImageNet query carries no trailing content at all — so the substituted instruction dangles, and accuracy collapses to 1.67, 0.00 and 0.00 respectively (gaps of 77.33, 52.33, 80.67, the three largest cells in the study). Even with the trio excluded, each stratum's mean swap gap (7.98 / 11.08 / 6.84) exceeds the same 33 tasks' mean paraphrase gap (2.33) by $2.9\times$ to $4.8\times$, so the ordering does not rest on the malformed cells. We therefore claim only the robustness result above, not a calibrated lexical-distance curve. Mean embedding displacement tells the same story from the other side: cosine to the original query embedding is essentially flat across strata (0.637 / 0.599 / 0.650, rank correlation with overlap -0.031) and well below the paraphrase condition's 0.78, so any wrong-task instruction moves the query about equally far regardless of how much wording it shares with the right one — what a task-identity signal would do, and not what graded surface-form noise would do.

Per-task swap numbers are donor-dependent and should be read as such. The gap's range across the four donors averages 15.81 points per task and reaches 76.33 (ImageNet-R: a gap of 4.34 under its farthest donor, 80.67 under its nearest). This does not touch the aggregate comparison, which is a difference of means over 36 tasks under a fixed donor rule, but it does mean no individual task's swap gap in Table 32 should be treated as *the* cost of a wrong instruction for that task. Table 9

gives the per-task cells; `code/swapgrade_report.py` reproduces every number here from `runs/perturb_swapgrade/`.

Table 9: Per-task swap gap under each donor, all 36 tasks, 300 queries each. Gaps are points below *orig*; positive means the substituted instruction hurt. Task order as in Table 32.

Task	orig	Δnear	Δmid	Δfar	Δhand
ImageNet-1K	79.00	77.33	18.00	15.67	32.00
N24News	74.33	15.33	14.33	22.33	28.66
HatefulMemes	60.67	7.00	5.34	6.67	13.67
VOC2007	74.00	31.00	22.33	31.00	20.67
SUN397	73.33	32.33	21.00	14.66	32.33
Place365	33.00	8.33	2.67	−0.33	2.67
ImageNet-A	52.33	52.33	15.00	11.33	52.33
ImageNet-R	80.67	80.67	23.67	4.34	30.00
ObjectNet	36.33	−12.00	0.00	−14.00	−12.00
Country211	21.00	18.67	13.67	6.33	18.67
OK-VQA	42.67	11.00	5.67	4.34	2.00
A-OKVQA	39.33	8.66	2.00	5.33	3.66
DocVQA	85.00	5.67	5.67	7.33	5.67
InfographicsVQA	42.67	8.00	7.34	7.34	7.34
ChartQA	44.00	17.00	16.00	15.67	14.00
Visual7W	46.67	5.34	9.34	14.34	7.34
ScienceQA	32.67	8.67	2.34	3.00	2.34
VizWiz	34.00	16.67	8.33	8.33	6.33
GQA	47.67	9.34	−16.33	−18.00	−14.66
TextVQA	65.00	13.33	6.67	6.00	6.67
VisDial	72.00	−1.67	−1.00	2.00	0.00
CIRR	47.67	20.67	41.67	22.00	20.67
VisualNews_t2i	72.00	0.67	0.67	2.33	2.33
VisualNews_i2t	74.00	14.33	70.67	12.67	13.00
MSCOCO_t2i	77.33	1.00	3.00	3.00	0.66
MSCOCO_i2t	67.67	21.34	49.00	9.00	19.34
NIGHTS	66.00	4.33	11.33	6.00	5.67
WebQA	87.67	1.00	5.34	4.00	5.34
FashionIQ	14.00	4.67	3.33	−1.67	4.67
Wiki-SS-NQ	52.00	2.33	4.33	0.00	0.00
OVEN	66.67	−4.00	5.34	1.00	2.00
EDIS	78.33	0.66	2.00	1.00	2.00
MSCOCO	63.00	6.33	21.67	21.00	13.00
RefCOCO	81.33	−3.67	21.00	11.33	3.66
RefCOCO-Matching	76.67	0.67	7.00	17.67	10.00
Visual7W-Pointing	68.67	−9.66	−6.00	−6.00	−3.00
mean	59.15	**13.16**	**11.73**	**7.14**	**9.97**

Two graded controls between paraphrase and swap. A single hand-picked wrong-task donor and three paraphrases span the semantic axis at only two points. Two further conditions, run on the same 36-task VLM2Vec-2B grid against the same cached index, fill in the middle and the far end. *minedit* replaces exactly one content word of the fixed instruction so that the task changes while more than 90% of the string survives — "classification" → "retrieval", "matches" → "contradicts", "object" → "action", "can answer" → "cannot answer" (`instructions.MINEDIT`, one edit per distinct MMEB string, drafted with LLM assistance and author-screened). A model reading instruction semantics should pay for the edit; a model reading surface form should pay about what a paraphrase costs or less, since almost every token is intact. *shuffle* permutes the words of the fixed instruction in a seeded random order with the image token and trailing connector kept, so length, vocabulary and layout match *orig* exactly while the semantics are destroyed; it is the format-matched control the swap donors lack.

Result: neither graded condition costs more than a paraphrase. Over the 36 tasks the macro gaps are paraphrase 3.32, *minedit* 2.99, *shuffle* 3.66, deletion 7.65 and swap 9.97 (`runs/perturb_pq_ext`, `code/ext_summary.py`; the run's own *orig* reproduces the paper's grid to the query). *minedit* costs more than the paraphrase mean on 19 of 36 tasks and *shuffle* on 19 of 36 (sign test $p = 0.87$ for both), whereas the swap costs more than *minedit* on 27 of 36 ($p = 0.004$) and more than *shuffle* on 30 of 36 ($p = 7 \times 10^{-5}$), and deletion costs more than *shuffle* on 31 of 36 ($p = 1.3 \times 10^{-5}$). Changing one word so that the stated task changes, or permuting every word so that no task is stated, costs what rewording the same task costs: ImageNet-1K's "for classification" $\rightarrow$ "for retrieval" costs 4.7 against a swap of 32.0, CIRR's "with" $\rightarrow$ "without the described changes" costs 0.3, and shuffling ImageNet-1K's string costs 12.7. The exceptions fit the same reading: for SUN397 and Place365 the edited word is the one that carries the label space ("scene" $\rightarrow$ "season"), and the edit costs 32.3 and 16.3. Stating the wrong task is therefore not by itself what makes a swap expensive: *minedit* states one and costs more than the paraphrase mean on only about half the tasks (19 of 36). What the swap adds cannot be attributed further here. Substituting another task's string changes at once the stated task, the vocabulary, whether the exact string was trained on, and how the query fits the candidate template, and no condition varies one of these alone (*minedit* keeps over 90% of the string, *shuffle* its exact vocabulary). Swap $>$ paraphrase $\approx$ *minedit* $\approx$ *shuffle* is an ordering, not a split of the swap gap into semantic and surface shares, and the separately trained token arms (Appendix B.14) supply no such split either: accuracies of different models are not components of one model's cost.

B.3 Paraphrase dispersion

Every audit number in the main grid is a point estimate over one fixed sample; per-query paired bootstrap intervals for the macro gaps are reported separately in Appendix B.4, once the grid was re-run storing per-query outcomes rather than only per-cell accuracies. Here we report a different and complementary quantity: the dispersion observed *across the three paraphrases*, the closest available proxy for how much a single wording choice moves a task's number. Across the 36 tasks the three paraphrase accuracies span a mean range of 4.57 points (median 3.34, max 28.33 on ImageNet-R) with mean standard deviation 1.98; that spread exceeds the mean paraphrase gap itself (3.32) on 21 of 36 tasks, and exceeds a task's own paraphrase gap on 20 of 36.

Two consequences follow, and we hold to both. Per-task gaps in Table 32 (Appendix B.23) should be read qualitatively, as the identity of the brittle tail rather than as calibrated effect sizes, particularly on the small-pool tasks (VOC2007: 20 candidates; HatefulMemes: 2). But the aggregate ordering the paper rests on is not a within-task-noise effect: the paraphrase/swap separation is a difference of means over 36 tasks, and the wording-to-wording spread measured here is a quantity an instruction-follower should keep small, since all three wordings mean the same thing. A benchmark whose reported score moves several points depending on which of three synonymous phrasings an author happened to pick is the finding, not an obstacle to it.

Does the ordering depend on which paraphrases we wrote? Two checks that need no rater. First, score each task under its *most favourable* paraphrase rather than the mean of three: the wrong-task instruction still costs more on 32 of 36 tasks (sign test $p = 1.9 \times 10^{-6}$; macro best-paraphrase gap 1.18). Second, drop any one of the three paraphrase slots from every task: the macro paraphrase gap moves between 3.02 and 3.84 and swap $>$ para holds on 30–31 of 36 tasks in every case. A single badly written paraphrase per task therefore cannot manufacture the ordering, though it can move a task's number (`code/grid_summary.py`).

Blinded ratings. Three colleagues who are not authors rated all 252 pairs (108 trained-on paraphrases, 108 held-out paraphrases, 36 swaps; order shuffled, task names and conditions hidden; sheet, instructions and analysis in `runs/rating/` and `code/rating_results.py`) on two 1–5 questions: Q1, do A and B ask for the same task; Q2, is B a well-formed instruction. A pair counts as accepted only if every rater scores it ≥ 4 on the question at issue; ratings were not adjudicated, so disagreements are reported as they stand. Two raters filled the sheet row by row; the third recorded their judgments as an explicit rule list (a default score with enumerated exceptions and comments) that was then expanded into the sheet. On Q1 the raters agree (Fleiss' $\kappa = 0.82$ over five categories, 1.00 once collapsed to same-task ≥ 4 versus not): 215 of 216 paraphrase pairs were scored 4 or 5 by every rater, and all 36 swap pairs were scored below 4 by every rater (34 unanimously ≤ 2). The one exception is the second held-out DocVQA paraphrase ("Which part of this document answers

the following? . . . "), which every rater read as grounding rather than joint encoding; it is also that task's largest held-out drop (7.0 points). Restricting the audit to unanimously accepted pairs leaves the trained-on ordering unchanged (paraphrase gap 3.32, swap gap 9.97, swap > para on 31 of 36) and moves the held-out gap from 3.33 to 3.27 (swap > para on 27 rather than 26 of 36). Q2 did not reach agreement ($\kappa \approx 0$): one rater scored 57 classification and pointing paraphrases as awkward (3) where the other two scored 5; requiring unanimity on both questions keeps 26 tasks, on which the ordering still holds (2.03 vs 6.60; 23 of 26).

B.4 Per-query paired bootstrap intervals

Appendix B.3 could not supply confidence intervals because the audit stored per-cell accuracies rather than per-query outcomes. We re-ran the full 36-task, 6-condition VLM2Vec-Qwen2VL-2B grid with per-query correctness and ranking margins retained (`perturb_eval.py --save_per_query`, `runs/perturb_pq`) and computed paired bootstrap intervals over queries (`code/bootstrap_report.py`, 10,000 resamples). Because every condition is scored on the same 300 queries per task, the resample is paired: one draw of query indices is applied to all six conditions at once, which is what makes an interval on a *difference* of gaps (not just a single gap) meaningful. We report two resampling schemes, because they answer different questions: *query* holds the 36-task set fixed and resamples only queries within each task, answering how much of the gap is query-sampling noise; *cluster* resamples tasks with replacement and then queries within each drawn task, the honest interval for a macro average taken over a population of tasks, and here it is the wider of the two for every quantity.

Table 10: Paired bootstrap 95% intervals for VLM2Vec-Qwen2VL-2B's macro gaps (point estimate first, then [query-resampled CI] / [cluster-resampled CI]).

Quantity	Point	Query-resampled 95% CI	Cluster-resampled 95% CI
orig	59.15	[58.30, 60.01]	[52.72, 65.26]
para_gap	3.32	[2.73, 3.90]	[0.69, 6.00]
para_worst_gap	5.75	[5.32, 6.69]	[2.90, 9.20]
none_gap	7.65	[6.88, 8.42]	[4.29, 11.02]
swap_gap	9.97	[9.19, 10.76]	[5.68, 14.56]
swap−para	6.65	[5.97, 7.34]	[3.92, 9.82]
swap/para	3.00	[2.61, 3.53]	[1.96, 8.69]

Every interval for `swap_gap − para_gap` excludes zero under both resampling schemes, and the swap gap exceeds the paraphrase gap in all 10,000 replicates under both schemes (a bootstrap frequency, not a probability) — the central ordering claim survives the honest, macro-over-tasks interval, not just the point estimate. The cluster interval is substantially wider than the query interval for every quantity (e.g. swap/para: [2.61, 3.53] against [1.96, 8.69]), which is expected and is exactly why we report it separately rather than only the narrower query interval: a reader asking "would this ratio survive a different sample of 36 MMEB tasks" should use the cluster column, not the query column.

We additionally re-ran the bootstrap on the two pool strata already reported in Appendix B.11 (pool $\geq$ 1,000, 26 tasks; pool $<$ 100, 3 tasks), using cluster resampling within each stratum. The large-pool stratum's swap−para difference is 5.66 points, cluster CI [2.62, 9.64], excluding zero, and its swap/para ratio is 3.07, CI [1.88, 9.20] — consistent with, and no tighter than, the all-task interval, since it is still only 26 tasks. The small-pool stratum's swap−para difference is 6.30 points, but its cluster CI is [−3.78, 16.30], crossing zero, and the ratio's own CI, [0.82, 3.31], crosses 1 as well: neither statistic supports an ordering in a 3-task stratum, which we read as the interval correctly reporting that such a stratum cannot carry an ordering claim on its own, not as evidence against the ordering, which is carried by the 26-task and all-task intervals.

The same paired bootstrap (10,000 replicates, per-query records re-run for every checkpoint) covers all five audited checkpoints, plus VLM2Vec-V2.0 scored on MMEB-V1 for the exposure comparison of Appendix B.6 (Table 11). For the three replications the cluster-resampled interval on swap−para excludes zero: VLM2Vec-7B 9.61 [5.35, 14.42], Qwen3-2B 4.57 [2.48, 7.10], Qwen3-8B 4.49 [1.75, 7.28]; the swap/para ratio's cluster interval stays above 1 for all three (7B [2.79, 16.87], Qwen3-2B

[1.34, 2.45], Qwen3-8B [1.34, 4.29]). For GME the difference is clear of zero only under query resampling (0.67 [0.14, 1.21]) and not under task resampling ([−3.74, 5.11]), and every GME gap is a fraction of the others', so the interval that matters for a checkpoint-level claim does not support an ordering there. For VLM2Vec-V2.0 on MMEB-V1 both differences clear zero under task resampling (swap−para 8.59 [3.85, 13.70], ratio 3.63 [2.18, 6.49]). Between-checkpoint difference intervals are left as future work.

Table 11: Cluster-resampled 95% intervals (tasks, then queries; 10,000 replicates) for every audited checkpoint. Point estimates as in Table 2 (VLM2Vec-V2.0 on MMEB-V1: Appendix B.6); Qwen3-2B's para_gap is 6.41 here against 6.40 there because its per-query rerun differs by one query on each of two tasks. Runs: `perturb_pq`, `perturb_7b_pq`, `qwen3ve2B_pq`, `qwen3ve8B_pq`, `perturb_gme_final`, `perturb_vlm2vecv2_pq`.

Checkpoint	para_gap	swap_gap
VLM2Vec-Qwen2VL-2B	3.32 [0.69, 6.00]	9.97 [5.68, 14.56]
VLM2Vec-Qwen2VL-7B	2.53 [0.39, 4.65]	12.14 [7.05, 17.81]
Qwen3-VL-Embedding-2B	6.41 [3.58, 9.60]	10.98 [7.36, 15.09]
Qwen3-VL-Embedding-8B	4.04 [1.69, 6.86]	8.53 [5.32, 12.31]
GME-Qwen2VL-2B	1.41 [0.17, 2.69]	2.07 [−1.33, 8.89]
VLM2Vec-V2.0 (MMEB-V1)	3.26 [1.35, 5.41]	11.85 [6.15, 18.05]

Checkpoint	swap−para	swap/para
VLM2Vec-Qwen2VL-2B	6.65 [3.92, 9.82]	3.00 [1.96, 8.69]
VLM2Vec-Qwen2VL-7B	9.61 [5.35, 14.42]	4.80 [2.79, 16.87]
Qwen3-VL-Embedding-2B	4.57 [2.48, 7.10]	1.71 [1.34, 2.45]
Qwen3-VL-Embedding-8B	4.49 [1.75, 7.28]	2.11 [1.34, 4.29]
GME-Qwen2VL-2B	0.67 [−3.74, 5.11]	1.47 [−0.64, 7.81]
VLM2Vec-V2.0 (MMEB-V1)	8.59 [3.85, 13.70]	3.63 [2.18, 6.49]

B.5 Full-query replication of the VLM2Vec-2B grid

Every grid in the paper scores each task's first 300 evaluation queries, a compute choice made when the audit grew to five checkpoints and several training arms. To check that the choice does no work, we rescored all six conditions of the VLM2Vec-Qwen2VL-2B grid on every evaluation query of every task (1,000 per task, 36,000 queries per condition) against the byte-identical candidate cache of `runs/perturb_pq` (`runs/perturb_full_pq`, `code/fullq_summary.py`). Three checks, in the order they matter.

The protocol is deterministic. The first-300-query slice of the full run reproduces the paper's grid to within one query per cell: over all 216 task×condition cells the largest difference is 0.33 points (one of 300 queries; EDIS/*orig*, flipped by a different batch composition). Any change below is therefore sampling, not protocol.

On all queries no macro-mean gap moves by more than half a point. Table 12: para_gap 3.32 → 3.69, worst-of-three 5.75 → 6.13, none_gap 7.65 → 7.70, swap_gap 9.97 → 9.54. Swap costs more than paraphrase on 32 of 36 tasks (sign test $p = 1.9 \times 10^{-6}$, Wilcoxon 4.7×10^{-6}) against 31 of 36 on the 300-query grid, and the cluster-resampled interval on swap−para, [3.52, 8.62], excludes zero as before. The swap/paraphrase ratio is 2.59 [1.80, 5.66] against 3.00 [1.96, 8.69], so the abstract's "a third" remains a rounded ordering statement either way. The 26-task large-pool stratum keeps the ordering on its own (swap−para 5.31 [2.69, 8.62]) and the 3-task small-pool stratum still cannot ([−2.06, 15.47]), as in Appendix B.4.

The 700 queries per task the paper never scored say the same thing. Restricted to queries 301–1,000: para 3.85, worst-of-three 6.43, swap 9.37, none 7.73; swap > para on 29 of 36 tasks ($p = 3.1 \times 10^{-4}$, Wilcoxon 2.1×10^{-5}); medians 1.86 (para) against 6.29 (swap); the worst task is again VOC2007 (25.2). This is the out-of-sample replication a query-sampling objection asks for.

Per task (Table 32, right-hand columns), the largest moves between the two grids fall on the small-sample classification tasks Appendix B.3 already flags as dispersed: ImageNet-R's swap gap 30.0 $\rightarrow$ 16.3, ImageNet-A's paraphrase gap 10.9 $\rightarrow$ 14.7, HatefulMemes 6.2 $\rightarrow$ 10.3, RefCOCO's swap gap 3.7 $\rightarrow$ 7.1. Nothing changes sign on the brittle tail (VOC2007 24.2 $\rightarrow$ 24.9, CIRR 20.6 $\rightarrow$ 20.9, ImageNet-1K 16.6 $\rightarrow$ 14.6) or at the inverted end (ObjectNet $-15.1 \rightarrow -17.0$, GQA $-12.2 \rightarrow -12.9$). We keep the 300-query grid as the paper's reference because every other analysis — the four other checkpoints, the side decomposition and the training arms — is paired to it; the full grid is the check that the pairing costs nothing.

Table 12: Full-query replication of the VLM2Vec-Qwen2VL-2B grid. Column 1 is the paper's grid (each task's first 300 queries, Table 10); columns 2–3 rescore all six conditions on every eval query (1,000 per task, same candidate cache) with paired bootstrap intervals (10,000 replicates); column 4 restricts to queries 301–1,000, which the paper's grid never scored. Cluster intervals resample tasks, then queries.

Quantity	300 queries [cluster CI]	all queries [query CI]
orig	59.15 [52.72, 65.26]	59.66 [59.19, 60.12]
para_gap	3.32 [0.69, 6.00]	3.69 [3.37, 4.00]
para_worst_gap	5.75 [2.90, 9.20]	6.13 [5.84, 6.60]
none_gap	7.65 [4.29, 11.02]	7.70 [7.27, 8.13]
swap_gap	9.97 [5.68, 14.56]	9.54 [9.11, 9.98]
swap−para	6.65 [3.92, 9.82]	5.86 [5.48, 6.24]
swap/para	3.00 [1.96, 8.69]	2.59 [2.41, 2.79]
swap > para (tasks)	31/36	32/36

Quantity	all queries [cluster CI]	queries 301–1000
orig	[53.50, 65.66]	59.88
para_gap	[1.11, 6.34]	3.85
para_worst_gap	[3.09, 9.66]	6.43
none_gap	[4.51, 10.99]	7.73
swap_gap	[5.71, 13.86]	9.37
swap−para	[3.52, 8.62]	5.52
swap/para	[1.80, 5.66]	2.43
swap > para (tasks)	$P = 1.000$	29/36

B.6 MMEB-V2 VISUAL-DOCUMENT AUDIT

MMEB-V2 (Meng et al., 2025) evaluates its visual-document retrieval datasets — in the released evaluation set, 27: ViDoRe v1 (10) and v2 (7) (Faysse et al., 2025), VisRAG (6), ViDoSeek (2) and MMLongBench (2) — with *one* query instruction, "Find a document image that matches the given query:", and *one* candidate template, "Understand the content of the provided document image.", shared by every dataset (VLM2Vec `vidore_dataset.py`, `visrag_dataset.py`). The convention audited on MMEB-V1 at one string per task therefore holds here at one string per meta-task. We ran the six query-side conditions of Section 3 on all 27 datasets: *orig*; three meaning-preserving paraphrases (drafted with LLM assistance and screened by the authors; AI use statement); *swap*, MMEB-V1's MSCOCO text-to-image instruction "Find me an everyday image that matches the given caption:" — a retrieval instruction for the wrong kind of image; and *none*. The candidate template is held at *orig* throughout, so as in Section 3 the candidate index is encoded once and reused byte-identically across conditions. Scoring follows MMEB-V2: nDCG@5 over the query's pool (the dataset's `corpus_range` where it defines one, otherwise the whole corpus), every query of every dataset, pages at the model's own resolution setting (`code/visdoc_instructions.py`, `code/visdoc_perturb_audit.py`). Three checkpoints: VLM2Vec-V2.0 (Meng et al., 2025), the model MMEB-V2 was released with (a DoRA adapter on Qwen2-VL-2B whose training data uses this instruction); Qwen3-VL-Embedding-2B, which reports MMEB-V2 results; and VLM2Vec-Qwen2VL-2B, the paper's main checkpoint, which was trained on MMEB-V1 only and never saw

the string — for it the visdoc instruction is a never-trained wording, the natural control for Section 4's exposure account.

Data notes. In the September-2026 MMEB-V2 release the VisRAG image files carry hash suffixes that cannot be derived from the corpus ids, SlideVQA ships no image directory, ViDoSeek pages are shipped once for both splits, and `esg_reports_human_labeled_v2` lists each of its 1,538 pages twice. We rebuilt the affected pages from the image bytes carried by the corpus parquets and deduplicated that corpus (`code/visdoc_fill_images.py`); no query, qrel or page content was altered. Two splits are saturated by construction, in the sense that every candidate in a query's pool is relevant: MMLongBench-doc and ViDoSeek-doc restrict each query's pool (`corpus_range`) to the pages of the target document and mark every one of them relevant (graded 2, with 3 for the evidence pages), so P@1 is 100 for any ranking and nDCG@5 measures only within-document ordering; both are nearly inert, every gap within 0.35 nDCG@5 points of zero under every condition and checkpoint, and we report them without excluding them. Macro means over the 25 remaining datasets are given alongside.

Qwen3-VL-Embedding-2B. Over the 27 datasets (12,917 queries) a paraphrase of the single shared string costs 2.22 nDCG@5 points on average (worst-of-three 5.51; query-resampled CI [1.86, 2.60], cluster-resampled [1.56, 2.94]), the wrong-task instruction 4.82 [3.30, 6.58] and deleting it 3.23 [2.22, 4.28]; swap costs more than paraphrase on 23 of 27 datasets (sign test $p = 3.1 \times 10^{-4}$) and the cluster interval on swap−para, [1.41, 3.99], excludes zero. The paraphrase gap is positive on 26 of 27 datasets (TabFQuAD, -0.16, is the exception); over the 25 non-saturated datasets the four gaps are 2.39 / 5.93 / 5.20 / 3.47 and the count 22 of 25. Sensitivity concentrates in ViDoRe v2 (paraphrase 3.38, swap 10.08 over its 7 datasets; the ESG-report corpora lose 5.3 and 4.8 points to a paraphrase and 19.8 and 13.1 to the swap), with VisRAG and ViDoRe v1 at 2.1 / 3.5–3.9 and the two document-level splits nearly inert. The ordering matches the same checkpoint's MMEB-V1 grid (Table 2: paraphrase 6.40, swap 10.98), whose gaps are larger in points; MMEB-V1 scores top-1 accuracy and this split nDCG@5, so we do not compare the magnitudes as ratios. A string shared by 27 datasets cannot identify a dataset, only the meta-task, so smaller gaps are what the task-ID reading expects here; we note the consistency without claiming the comparison as a test.

VLM2Vec-V2.0. The checkpoint MMEB-V2 was released with is close to instruction-insensitive on this meta-task. Over the 27 datasets (12,917 queries) a paraphrase costs 0.57 nDCG@5 points (worst-of-three 1.36; query CI [0.33, 0.82], cluster CI [0.20, 0.99]), the wrong-task instruction 1.16 [0.65, 1.69], and deleting the instruction 2.33 [1.57, 3.15] — more than substituting a wrong one, on 18 of 27 datasets. The ordering of Section 3 still holds (swap $>$ para on 21 of 27 datasets, $p = 0.006$; swap−para cluster interval [0.15, 1.02]), but at a quarter of Qwen3-2B's magnitudes; the paraphrase gap is positive on 21 of 27 datasets, no dataset loses more than 3.3 points to a paraphrase (Economics reports) or 3.9 to the swap, and over the 25 non-saturated datasets the gaps are 0.62 / 1.47 / 1.24 / 2.51 (19 of 25). That deletion costs more than substitution says what this model reads from the shared string: that an instruction is present, not which one.

VLM2Vec-Qwen2VL-2B: a wording it never trained on. The paper's main checkpoint was trained on MMEB-V1 only, so the visdoc instruction is a string it has never seen and its absolute scores are low (orig 41.28). Here a paraphrase costs nothing: the mean paraphrase gap is -0.75 (query CI [-1.17, -0.34], cluster CI [-1.54, -0.01]; positive on 10 of 27 datasets), while the wrong-task instruction still costs 1.15 [0.15, 2.22], swap $>$ para holds on 20 of 27 ($p = 0.019$; swap−para cluster interval [0.75, 3.16]), and deleting the instruction costs 0.40 [-0.75, 1.62], indistinguishable from zero. This is a never-trained-wording control at benchmark scale, which within MMEB-V1 only the nine unseen-string tasks supply (Appendix B.28): without exposure to the exact wording the wording has no privileged status, and what remains is the cost of a wrong instruction.

Reading. Only the checkpoint that never saw the string pays nothing for a paraphrase: V1-2B's paraphrase and wrong-instruction gaps are -0.75 and 1.15, VLM2Vec-V2.0's, trained with the string, 0.57 and 1.16, and Qwen3-2B's 2.22 and 4.82. Swap $>$ para holds for all three (20, 21 and 23 of 27 datasets; every cluster interval on swap−para excludes zero), with gaps smaller in points than the same checkpoints' MMEB-V1 gaps (V1-2B 3.32 / 9.97, Qwen3-2B 6.40 / 10.98; Table 2), which are top-1 accuracies. The contrast also appears within one checkpoint: scored on MMEB-V1's 36 tasks, which use 23 distinct per-task strings, VLM2Vec-V2.0 loses 3.26 to a paraphrase and 11.85 to the swap (swap $>$ para on 28 of 36, $p = 0.0012$; swap−para cluster interval [3.85, 13.70]; `runs/perturb_vlm2vecv2_pq`, Table 11), against 0.57 and 1.16 here. The weights are the

same, but the two benchmarks differ in more than whether the string is fixed per dataset or shared by 27: query modality, pool construction and metric change too. A string shared by 27 datasets identifies a meta-task, not a dataset, so the task-ID reading predicts less to be gained from its surface form; the smaller gaps are consistent with that, and we report the contrast rather than claim it as a test. What carries over unchanged is the blind spot: MMEB-V2 scores its visual-document datasets under one wording and, for the checkpoint it was released with, would report a number within about a point under any of the four alternatives we tried — a score that cannot tell following the instruction from ignoring it.

Seven wordings, not four. We also scored a second triple of paraphrases and MMEB-V1's Wiki-SS-NQ instruction, "Find the document image that can answer the given query:" — a near-verbatim rewording from another benchmark, absent from the MMEB-V1 training split. The second triple costs Qwen3-2B 1.47 [0.88, 2.08] against the first triple's 2.22, VLM2Vec-V2.0 0.10 [-0.35, 0.52] against 0.57, and V1-2B -1.03 [-1.70, -0.29] against -0.75; the borrowed wording scores *above* orig for Qwen3-2B (-0.43 [-0.90, -0.05], 22 of 27 datasets) and V1-2B (-1.16 [-1.77, -0.59], 24 of 27) and within noise for V2.0 (0.18 [-0.12, 0.48]). Which meaning-preserving wording a benchmark fixes therefore moves a checkpoint's reported number by up to 2.6 points and can flip the sign of its "paraphrase gap" — Appendix B.3's finding at meta-task scale (`runs/visdoc_*/*__hpara*`, `*__swapnear*`).

Table 13: MMEB-V2 visual-document audit, macro means over datasets (nDCG@5 points). Gaps are *orig* minus the condition; last column counts datasets where the wrong-task instruction costs more than a paraphrase (sign test p). Per-dataset values in Tables 14–16.

Checkpoint	Datasets	orig	para	worst	swap	none	swap>para
VLM2Vec-V2.0	all (27)	66.25	0.57	1.36	1.16	2.33	21/27 (5.9×10^{-3})
	ViDoRe v1 (10)	75.72	0.69	1.70	1.19	1.89	7/10
	ViDoRe v2 (7)	47.63	0.63	1.70	1.36	4.08	5/7
	VisRAG (6)	79.58	0.74	1.25	1.45	2.56	5/6
	ViDoSeek (2)	56.64	0.16	0.40	0.24	−0.36	2/2
	MMLongBench (2)	53.72	−0.27	−0.18	0.36	0.35	2/2
Qwen3-VL-Emb-2B	all (27)	73.25	2.22	5.51	4.82	3.23	23/27 (3.1×10^{-4})
	ViDoRe v1 (10)	83.22	2.14	4.98	3.47	2.90	8/10
	ViDoRe v2 (7)	60.18	3.38	8.68	10.08	5.55	7/7
	VisRAG (6)	83.77	2.12	5.65	3.92	2.68	6/6
	ViDoSeek (2)	56.07	0.63	1.20	0.17	1.05	0/2
	MMLongBench (2)	54.73	0.47	1.01	0.53	0.64	2/2
VLM2Vec-V1-2B	all (27)	41.28	−0.75	0.64	1.15	0.40	20/27 (0.02)
	ViDoRe v1 (10)	46.38	−1.07	0.40	0.32	0.50	6/10
	ViDoRe v2 (7)	24.50	−2.27	−0.52	1.85	−0.85	5/7
	VisRAG (6)	43.57	1.15	2.69	2.30	2.07	5/6
	ViDoSeek (2)	56.10	−0.04	0.36	0.12	−0.03	2/2
	MMLongBench (2)	52.73	−0.21	0.07	0.37	−0.34	2/2

Table 14: MMEB-V2 visual-document audit, VLM2Vec-V2.0: nDCG@5 under the benchmark's single fixed instruction (*orig*) and the four gaps below it (points; positive = the condition hurt). One instruction string is shared by all 27 datasets, so each column is one rewording applied benchmark-wide. $|Q|$ = queries scored, $|C|$ = pages in the pool.

Family	Dataset	$\|Q\|$	$\|C\|$	orig	Δpara	Δworst	Δswap	Δnone
ViDoRe v1	ArxivQA	500	500	81.48	1.03	1.42	0.96	2.38
	DocVQA	451	500	45.94	1.05	2.10	1.51	2.99
	InfoVQA	494	500	84.01	0.76	1.14	1.75	5.04
	Shift Project	100	999	60.87	0.72	3.17	0.89	0.34
	SynthDocQA-AI	100	968	87.13	−0.46	−0.02	0.48	0.24
	SynthDocQA-Energy	100	975	86.25	0.70	2.15	0.44	0.95
	SynthDocQA-Gov	100	972	84.44	1.27	2.53	0.99	2.10
	SynthDocQA-Health	100	963	93.18	1.16	1.60	2.85	1.99
	TAT-DQA	1646	271	44.72	0.20	0.96	0.31	1.59
	TabFQuAD	280	70	89.20	0.41	1.94	1.68	1.30
ViDoRe v2	Biomedical (en)	160	1016	51.84	−0.45	0.98	−0.50	1.86
	Biomedical (multi)	640	1016	46.83	−0.78	0.71	0.57	2.76
	ESG human-labeled	104	1538	51.11	0.46	1.84	−0.80	6.46
	ESG reports (en)	57	1538	40.00	0.19	1.54	2.55	3.26
	ESG reports (multi)	228	1538	37.09	−0.41	0.05	1.55	4.35
	Economics (en)	58	452	56.74	3.29	4.24	3.94	4.66
	Economics (multi)	232	452	49.81	2.08	2.51	2.18	5.24
VisRAG	ArxivQA	816	8066	77.50	0.61	0.86	0.13	3.09
	ChartQA	63	500	82.23	0.66	1.32	1.20	3.38
	InfoVQA	718	459	86.45	1.27	1.92	2.12	2.99
	MP-DocVQA	591	741	72.72	0.52	0.75	1.87	1.86
	PlotQA	863	9593	66.28	0.39	1.12	1.55	1.62
	SlideVQA	556	1284	92.33	0.96	1.55	1.83	2.40
ViDoSeek	ViDoSeek (doc)	1142	5349	89.02	0.04	0.10	0.06	−0.09
	ViDoSeek (page)	1142	5349	24.27	0.28	0.70	0.41	−0.63
MMLongBench	MMLongBench (doc)	838	6492	85.89	−0.10	−0.08	0.11	0.08
	MMLongBench (page)	838	6492	21.54	−0.43	−0.29	0.62	0.62
Macro mean over 27 datasets				66.25	0.57	1.36	1.16	2.33
swap > para on 21/27 datasets (sign test $p = 5.9\times10^{-3}$)								

Table 15: MMEB-V2 visual-document audit, Qwen3-VL-Emb-2B: nDCG@5 under the benchmark's single fixed instruction (*orig*) and the four gaps below it (points; positive = the condition hurt). One instruction string is shared by all 27 datasets, so each column is one rewording applied benchmark-wide. $|Q|$ = queries scored, $|C|$ = pages in the pool.

Family	Dataset	$\|Q\|$	$\|C\|$	orig	Δpara	Δworst	Δswap	Δnone
ViDoRe v1	ArxivQA	500	500	80.62	3.29	6.85	2.94	3.43
	DocVQA	451	500	47.74	4.94	8.28	4.41	3.53
	InfoVQA	494	500	86.96	0.53	2.65	1.76	1.41
	Shift Project	100	999	80.56	0.51	1.47	4.33	0.48
	SynthDocQA-AI	100	968	96.18	2.08	6.00	4.12	3.59
	SynthDocQA-Energy	100	975	91.34	1.49	5.04	1.80	1.87
	SynthDocQA-Gov	100	972	95.92	2.37	5.80	2.84	3.62
	SynthDocQA-Health	100	963	97.02	2.16	3.85	5.09	3.59
	TAT-DQA	1646	271	61.37	4.16	9.13	6.36	6.98
	TabFQuAD	280	70	94.54	−0.16	0.74	1.02	0.45
ViDoRe v2	Biomedical (en)	160	1016	62.90	2.80	7.05	5.30	5.85
	Biomedical (multi)	640	1016	62.33	4.15	8.29	7.43	7.01
	ESG human-labeled	104	1538	58.96	5.27	10.88	19.80	6.29
	ESG reports (en)	57	1538	55.48	3.82	10.13	9.71	7.64
	ESG reports (multi)	228	1538	57.54	4.75	12.11	13.12	8.67
	Economics (en)	58	452	64.57	1.45	5.76	8.31	1.21
	Economics (multi)	232	452	59.50	1.43	6.57	6.87	2.18
VisRAG	ArxivQA	816	8066	80.84	2.77	5.89	4.27	3.35
	ChartQA	63	500	82.87	2.70	10.39	3.54	0.55
	InfoVQA	718	459	91.21	0.83	3.47	4.04	1.91
	MP-DocVQA	591	741	84.95	3.60	7.11	4.26	6.07
	PlotQA	863	9593	68.95	1.47	3.83	3.98	1.72
	SlideVQA	556	1284	93.77	1.36	3.19	3.40	2.45
ViDoSeek	ViDoSeek (doc)	1142	5349	88.87	0.16	0.30	0.04	0.26
	ViDoSeek (page)	1142	5349	23.27	1.11	2.10	0.29	1.83
MMLongBench	MMLongBench (doc)	838	6492	86.23	0.16	0.34	0.17	0.23
	MMLongBench (page)	838	6492	23.22	0.79	1.67	0.89	1.04
Macro mean over 27 datasets				73.25	2.22	5.51	4.82	3.23
swap $>$ para on 23/27 datasets (sign test $p = 3.1\times10^{-4}$)								

Table 16: MMEB-V2 visual-document audit, VLM2Vec-V1-2B: nDCG@5 under the benchmark's single fixed instruction (*orig*) and the four gaps below it (points; positive = the condition hurt). One instruction string is shared by all 27 datasets, so each column is one rewording applied benchmark-wide. $|Q|$ = queries scored, $|C|$ = pages in the pool.

Family	Dataset	$\|Q\|$	$\|C\|$	orig	Δpara	Δworst	Δswap	Δnone
ViDoRe v1	ArxivQA	500	500	42.02	0.42	3.95	0.35	−0.87
	DocVQA	451	500	26.88	−0.08	2.22	−0.40	−1.58
	InfoVQA	494	500	62.58	0.35	1.33	1.03	0.29
	Shift Project	100	999	44.67	−1.04	0.27	−3.06	4.49
	SynthDocQA-AI	100	968	55.14	−2.00	−1.29	0.51	−2.57
	SynthDocQA-Energy	100	975	49.16	−1.39	0.46	3.63	2.21
	SynthDocQA-Gov	100	972	52.30	−4.62	−3.72	1.19	−1.35
	SynthDocQA-Health	100	963	57.09	−1.84	−0.91	2.12	1.19
	TAT-DQA	1646	271	18.18	−0.77	−0.47	0.47	−0.09
	TabFQuAD	280	70	55.82	0.25	2.19	−2.61	3.33
ViDoRe v2	Biomedical (en)	160	1016	27.12	−4.39	−2.12	7.10	−1.97
	Biomedical (multi)	640	1016	20.84	−4.30	−1.88	3.02	−1.07
	ESG human-labeled	104	1538	19.45	−1.81	0.26	−2.07	−1.24
	ESG reports (en)	57	1538	28.46	0.18	2.57	5.84	4.34
	ESG reports (multi)	228	1538	24.34	−1.54	−0.69	1.57	4.26
	Economics (en)	58	452	27.31	−1.99	−0.93	−2.09	−5.36
	Economics (multi)	232	452	23.96	−2.05	−0.88	−0.42	−4.92
VisRAG	ArxivQA	816	8066	31.46	0.49	1.69	1.53	0.34
	ChartQA	63	500	44.04	2.11	3.12	5.37	6.98
	InfoVQA	718	459	62.43	0.98	3.69	1.10	0.50
	MP-DocVQA	591	741	40.70	1.94	3.89	1.91	1.69
	PlotQA	863	9593	11.59	0.36	0.56	2.11	2.19
	SlideVQA	556	1284	71.23	1.00	3.21	1.79	0.70
ViDoSeek	ViDoSeek (doc)	1142	5349	88.88	−0.01	0.09	0.03	−0.01
	ViDoSeek (page)	1142	5349	23.33	−0.07	0.64	0.20	−0.05
MMLongBench	MMLongBench (doc)	838	6492	85.62	−0.05	0.04	0.10	−0.09
	MMLongBench (page)	838	6492	19.85	−0.37	0.10	0.64	−0.60
Macro mean over 27 datasets				41.28	−0.75	0.64	1.15	0.40
swap $>$ para on 20/27 datasets (sign test $p = 0.02$)								

B.7 IMAGE-SIDE AUDIT: WHAT THE VISUAL CHANNEL CARRIES

Every other condition in the paper rewrites text. The query image is the one input all of them hold fixed, so none of them can say how much of a task's score the image carries, nor whether the paraphrase cost falls where the model leans on the image most or least. Two query-side conditions answer this on the paper's VLM2Vec-Qwen2VL-2B grid, with the instruction and the query's own text held byte-identical to *orig* and the candidate index unchanged (`code/perturb_eval.py`, conditions `blankimg` and `mismatchimg`; results `runs/perturb_pq_img`):

- *blank image* — the query image is replaced by a uniform mid-grey image (the evaluation pipeline resizes every image to 1344×1344 before encoding, so the placeholder is that size). This is the blind baseline of the VQA literature: what instruction and query text recover on their own. We call *vis* = orig − blank-image accuracy the task's *visual reliance*.
- *mismatched image* — the query image is replaced by another query's image from the same task, under a seeded derangement (no query keeps its own image; the permutation is saved with the run). This is the visual analogue of *swap*: a wrong image rather than no image. Where the first 300 queries of a task cover few targets the donor can carry the query's own target, so for those tasks the mismatched image is sometimes a match; Table 18 reports the rate.

Of the 36 tasks, 30 have evaluation queries that carry an image and are scored on the same first 300 queries as the paper's grid. VisDial, MSCOCO_t2i, VisualNews_t2i and Wiki-SS-NQ are text-query tasks, and WebQA's and EDIS's instruction templates carry an image token that no evaluation query fills (the pipeline drops it, as the official evaluation does), so these six have no image to remove. The run's own *orig* condition reproduces the paper's grid cell for cell (largest difference 0.00 over the 30 tasks), so every difference below is the image. Three predictions were written in the research log before any cell had run (not a public preregistration): (P1) a blank image costs far more than any text-side condition; (P2) the paraphrase gap is largest on the tasks whose visual reliance is highest, because on classification tasks the instruction is the whole of the query's text; (P3) a wrong image costs at least what no image costs.

The image carries most of the score, far more than any instruction condition (P1 confirmed). Over the 30 tasks a blank image costs 48.0 points on average, against 3.8 for a paraphrase, 8.6 for deleting the instruction and 11.6 for a wrong one (Table 17); it costs more than a wrong instruction on 27 of 30 tasks and more than deletion on 29. The query embedding moves accordingly: mean cosine to *orig* 0.38 under a blank image, against 0.75 under a paraphrase and 0.60 with the instruction deleted. Eight of the ten classification tasks fall to 0–3.3% with a blank image, since every query then becomes the same constant vector; the two that do not are N24News, whose query carries a headline (28.3), and HatefulMemes, a two-way task at chance (52.7). VQA tasks fall to a language-prior level of 2–34% and grounding tasks to 0–6%. The tasks that keep most of their score without an image are the two composed-retrieval tasks, where the modification text does most of the work: CIRR keeps 37.7 of 47.7 points (vis 10.0) and FashionIQ 6.7 of 14.0 (vis 7.3). This is a property of those benchmarks, not of the instruction, and GQA (34.3 of 47.7) is the one VQA task with a comparably strong language prior.

Brittleness concentrates in the high-reliance tasks, but reliance does not predict it (P2 partly supported). Over all 30 tasks the paraphrase gap correlates weakly and non-significantly with visual reliance (Spearman $\rho = +0.25$, permutation $p = 0.18$; swap gap $\rho = +0.32$, $p = 0.08$). Split at the median reliance (49.2), the high half loses 5.9 points to a paraphrase and 16.2 to a swap, the low half 1.7 and 7.0; within classification $\rho = +0.75$ ($p = 0.016$). The sharper reading is at matched reliance (Figure 3). Thirteen tasks lose 63–81 points to a blank image; among them the four classification tasks lose 11–24 points to a paraphrase (VOC2007 24.2, ImageNet-1K 16.6, ImageNet-R 15.5, SUN397 11.1) and the nine retrieval, grounding and VQA tasks lose at most 4.0 (MSCOCO_i2t), with DocVQA at 0.1 and RefCOCO at -0.3. What separates them is not how much they use the image but whether the instruction is the whole of the query's text: a classification query is image plus instruction and nothing else, so the instruction's surface form is the entire text the encoder reads. The counter-case runs the other way and makes the same point: CIRR uses the image least of any brittle task (vis 10.0) and still loses 20.6 points to a paraphrase. The alternative reading we wrote down in advance, that the instruction stands in for a neglected image, does not survive: the brittle tail is not the low-reliance tail.

Table 17: Image-side conditions beside the text-side ones, VLM2Vec-Qwen2VL-2B, the 30 MMEB-V1 tasks whose evaluation queries carry an image, 300 queries each. Every gap is points below *orig*, macro-averaged over the tasks in the row. *vis* = orig − blank-image accuracy (instruction and query text unchanged); *mis* = orig − mismatched-image accuracy. Sign tests over tasks: vis > swap on 27/30 ($p < 10^{-3}$), mis > vis on 12/30 ($p = 0.36$), vis > none on 29/30 ($p < 10^{-3}$). Spearman rank correlation with vis over tasks: paraphrase gap $\rho = +0.25$ (permutation $p = 0.18$), swap gap $\rho = +0.32$ ($p = 0.083$), none gap $\rho = +0.34$ ($p = 0.067$).

			text-side gap			image-side gap	
Tasks	n	orig	para	swap	none	vis	mis
all image-query tasks	30	56.33	3.81	11.62	8.59	48.01	45.02
classification	10	58.47	8.10	21.90	13.53	49.70	40.97
VQA	10	47.97	0.37	4.07	4.90	37.80	36.67
retrieval (image-query)	6	56.00	4.95	10.89	6.78	48.28	49.11
grounding	4	72.42	−0.03	5.91	8.17	68.92	69.92
8-task subset	8	74.42	12.67	21.71	19.66	64.71	55.12

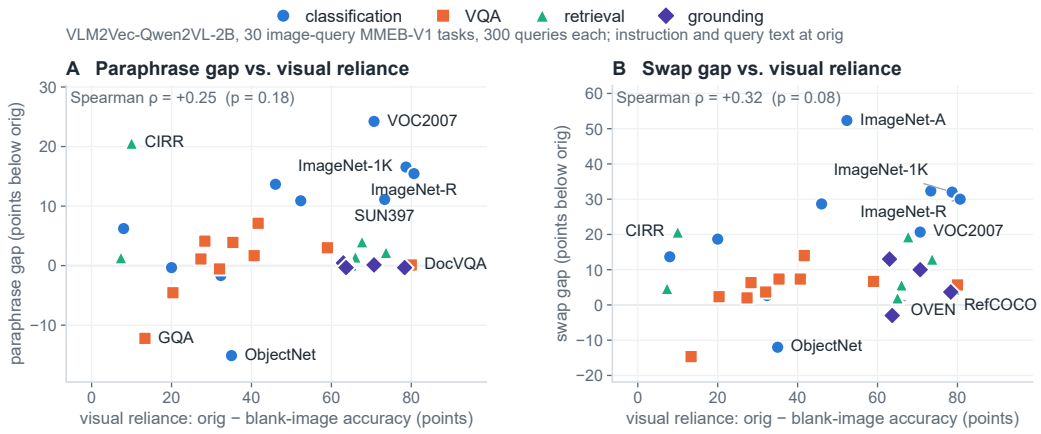


Figure 3: Paraphrase gap (A) and swap gap (B) against visual reliance, VLM2Vec-Qwen2VL-2B, the 30 image-query MMEB-V1 tasks. Visual reliance is *orig* minus blank-image accuracy with the instruction and the query text unchanged. Colour and marker shape give the MMEB category; the brittle and inverted tails are named. Among the thirteen tasks at matched reliance (63–81 points, right-hand cluster) the paraphrase gap runs from −0.3 to 24.2.

A wrong image costs about what no image costs (P3 not confirmed). Macro 45.0 against 48.0; a wrong image costs more than a blank one on 12 of 30 tasks ($p = 0.36$). The direction is task-specific in both senses. On ImageNet-R, ImageNet-A, VOC2007 and HatefulMemes the first 300 queries cover 3, 9, 20 and 2 targets, so 17–49% of donor images carry the query's own target and the mismatched-image accuracies (35.7, 13.3, 11.3, 51.0) are partly matches. On N24News a wrong but real news photograph recovers 60.7 against 28.3 for a blank: the headline carries the class, and a real image restores the input distribution a grey square breaks. On GQA the reverse holds, a wrong image (23.3) misleads where a blank (34.3) merely uninforms. We therefore do not read `mismatchimg` as a calibrated visual analogue of *swap*; it is a second bound. Whichever way the query image is spoiled, the cost is 45–48 points, an order of magnitude above a paraphrase and about four times a wrong instruction (3.9× for a wrong image, 4.1× for a blank one).

What this adds. The six text-side conditions could not say how the instruction effect relates to the image. The image carries most of the score in every task family, the paraphrase and swap costs sit on top of that at every level of reliance, and the tasks where a rewording costs most are those where the instruction is the only text the query carries, not those where the image is used least. We did not cross

image and instruction perturbations, so their interaction is untested; on VLM2Vec-Qwen2VL-2B, the only checkpoint measured here, the brittleness is best read as a property of the text channel.

Table 18: Per-task image-side audit, VLM2Vec-Qwen2VL-2B, 300 queries per task. Accuracy under *orig*, under a blank (uniform grey) query image and under another query's image from the same task; Δ = *orig* minus the condition (points; positive means the condition hurt). *same* is the share of queries whose donor image carried the query's own target, i.e. the mismatch was a match. The text-side columns repeat Table 32. The six tasks whose evaluation queries carry no image are omitted.

			blank image		mismatched image			text-side (Table 32)		
Task	cat.	orig	acc	Δ	acc	Δ	same	Δpara	Δswap	Δnone
ImageNet-1K	CLS	79.00	0.33	78.67	0.33	78.67	0%	16.56	32.00	23.67
N24News	CLS	74.33	28.33	46.00	60.67	13.66	5%	13.66	28.66	24.66
HatefulMemes	CLS	60.67	52.67	8.00	51.00	9.67	49%	6.22	13.67	5.67
VOC2007	CLS	74.00	3.33	70.67	11.33	62.67	17%	24.22	20.67	35.33
SUN397	CLS	73.33	0.00	73.33	0.33	73.00	0%	11.11	32.33	15.66
Place365	CLS	33.00	0.67	32.33	0.00	33.00	0%	−1.67	2.67	3.33
ImageNet-A	CLS	52.33	0.00	52.33	13.33	39.00	18%	10.89	52.33	13.00
ImageNet-R	CLS	80.67	0.00	80.67	35.67	45.00	44%	15.45	30.00	17.00
ObjectNet	CLS	36.33	1.33	35.00	2.00	34.33	2%	−15.11	−12.00	−13.34
Country211	CLS	21.00	1.00	20.00	0.33	20.67	1%	−0.33	18.67	10.33
OK-VQA	VQA	42.67	15.33	27.34	14.33	28.34	0%	1.12	2.00	5.67
A-OKVQA	VQA	39.33	7.33	32.00	8.33	31.00	0%	−0.56	3.66	1.66
DocVQA	VQA	85.00	5.00	80.00	5.33	79.67	0%	0.11	5.67	8.33
InfographicsVQA	VQA	42.67	7.33	35.34	10.67	32.00	0%	3.89	7.34	6.67
ChartQA	VQA	44.00	2.33	41.67	13.67	30.33	1%	7.11	14.00	16.33
Visual7W	VQA	46.67	6.00	40.67	8.33	38.34	1%	1.67	7.34	8.34
ScienceQA	VQA	32.67	12.33	20.34	19.00	13.67	0%	−4.55	2.34	0.00
VizWiz	VQA	34.00	5.67	28.33	4.67	29.33	12%	4.11	6.33	5.67
GQA	VQA	47.67	34.33	13.34	23.33	24.34	0%	−12.22	−14.66	−10.66
TextVQA	VQA	65.00	6.00	59.00	5.33	59.67	0%	3.00	6.67	7.00
CIRR	RET	47.67	37.67	10.00	37.00	10.67	0%	20.56	20.67	26.34
VisualNews_i2t	RET	74.00	0.33	73.67	0.33	73.67	0%	2.22	13.00	5.33
MSCOCO_i2t	RET	67.67	0.00	67.67	0.00	67.67	0%	4.00	19.34	7.34
NIGHTS	RET	66.00	0.00	66.00	0.00	66.00	0%	1.44	5.67	0.33
FashionIQ	RET	14.00	6.67	7.33	3.00	11.00	0%	1.33	4.67	4.00
OVEN	RET	66.67	1.67	65.00	1.00	65.67	1%	0.11	2.00	−2.66
MSCOCO	GND	63.00	0.00	63.00	1.33	61.67	0%	0.45	13.00	14.00
RefCOCO	GND	81.33	3.00	78.33	3.67	77.66	0%	−0.34	3.66	6.33
RefCOCO-Matching	GND	76.67	6.00	70.67	1.33	75.34	0%	0.12	10.00	11.67
Visual7W-Pointing	GND	68.67	5.00	63.67	3.67	65.00	0%	−0.33	−3.00	0.67

B.8 Curated-pool difficulty equivalence (exploratory)

On the 10 image-target tasks where candidate identity is free of the string-equivalence false negatives below, we take each task's accuracy under GT plus m uncurated random candidates drawn once, without replacement and without deduplication against semantically related candidates, from the shared 42,833-image index, at $m \in \{1000, 4000, 16000, 42832\}$, from a single seeded draw with no repetitions to average over. We interpolate that four-point curve, linearly in $\log m$, to the $m^\star$ at which random-pool accuracy equals the task's accuracy under its own officially curated pool (`code/pool_difficulty_equivalence.py`). On 9 of 10 tasks, $m^\star$ exceeds the curated pool's own size by 2×–28× (mean 12.1×), suggesting that MMEB accuracy at any pool size a deployed system would actually face depends substantially on how the benchmark's negatives were selected, and not on retrieval quality alone. The log-linear interpolation is an assumption, not a fitted and validated model, and the multiplier should be read as an order of magnitude rather than a measurement; on this evidence we do not claim the stronger form, that MMEB is principally

measuring negative-mining skill, which would need repeated draws and a validated curve. The exception, Wiki-SS-NQ, inverts the pattern — its curated pool is easier than 1,000 random negatives — while independently having the corpus's single worst targeted cross-task confusion score, indicating that pool difficulty and cross-task confusability are separate axes.

B.9 Systematic false negatives in the shared index (exploratory)

Scanning all 15,026 universal-shared-index top-1 errors across the 20 text-target tasks (20,000 queries), 16.9% are exact-string false negatives (15.3% without HatefulMemes, whose "Yes"/"No" answers recur verbatim among ChartQA's and DocVQA's candidates; most others are class-label strings shared across tasks, e.g. ImageNet-1K and ImageNet-A), and a further 12.1% are heuristic partial-match false negatives (e.g. GQA's "The plate is square." scored wrong against Visual7W's "Square."), for 29.0% combined. *Exact* means byte-identical after case-folding and punctuation stripping; *partial* means the normalized predicted and gold strings are in a substring relationship, or their token sets have Jaccard overlap ≥ 0.5, whichever is looser.

Both heuristics flag a false negative only on plain text overlap, so they miss paraphrase and world-knowledge equivalence (e.g. "photograph of a dog" vs. "canine") and can also flag near-matches that are genuinely different answers; the 29.0% combined figure is therefore a heuristic count that bounds the true false-negative rate from neither side, not an estimate of the heuristic's precision or recall against human judgment. We have not hand-verified a labeled subset to report those numbers. On that basis we do not present the implied correction — raw universal-index accuracy 24.87% rising to 37.55% (exact-only) or 46.66% (exact+partial) — as a validated restatement of benchmark accuracy; it indicates the order of magnitude at stake in an uncalibrated heuristic and should be treated as exploratory until a hand-labeled subset calibrates it. We therefore restrict our shared-index claims to the 10 false-negative-safe image-target tasks, where candidate identity is an image path, which rules out string-equivalence false negatives but not near-duplicate or semantically equivalent images.

B.10 Task-identity variance decomposition

The first row of Table 33 is a one-way between-group variance ratio, computed separately for queries and candidates. Let X be the stacked unit-norm embeddings, $\bar{x}$ their grand mean, and $\bar{x}_t$ the mean of the n_t embeddings belonging to task t; we report

$$\eta^2 = \sum_t n_t \|\bar{x}_t - \bar{x}\|^2 \Big/ \sum_i \|x_i - \bar{x}\|^2, \tag{1}$$

over all 36 tasks, on the query embeddings of the main audit and on the full shared candidate index respectively (`code/analyze_geometry.py`).

It is a descriptive decomposition, not a fitted model: there are no random effects, no per-factor terms for instruction, modality, and content, and — the important caveat — **instruction and content are not independently varied**. Because each MMEB task also has its own image and text distribution and its own target type, a large η^2 is consistent with task identity being recoverable from content alone, with no contribution from the template. This row therefore establishes that embeddings are strongly task-clustered, not that the instruction is what clusters them. The rows that *do* hold content fixed while moving only the template are the one-character-edit rows (identical images and identical candidate content, a single period removed, cosine 0.7397 against a 0.9994 same-string noise floor, a 3.5-point accuracy cost) and the SUN397 row; those, not the variance share, carry the template-specific weight of the argument. A decomposition that separates instruction, modality, task, and content as crossed factors would require an evaluation grid in which templates are assigned independently of task, which MMEB does not provide and we did not construct (Section 7).

B.11 Pool-size strata and query weighting

Per-task pool size is measured, not assumed: the task's candidate count is the row count of its cached candidate matrix and the per-query local pool size is the mean successive difference of its `local_offsets` index, both read from the evaluation cache (`code/pool_stratified_report.py`, artifact `runs/pool_stratified.json`). Strata boundaries are round decades fixed before looking at the gaps.

Table 19: The audit re-aggregated by measured candidate-pool size. Gaps are points below *orig* (positive = the condition hurt). *Mean pool* is the per-query candidate pool, 1,000 on every large-stratum task even where the task's candidate set is larger. Every cell in every stratum was scored on $n = 300$ queries, so query-weighted means equal macro means exactly.

Pool stratum	Tasks	Mean pool	para_gap	swap_gap	swap/para
tiny (< 10)	1	2.0	6.22	13.67	2.20
small (10–99)	2	22.0	18.94	24.66	1.30
medium (100–999)	7	416.6	0.62	11.10	17.90
large (1,000)	26	1000.0	2.74	8.40	3.07
all tasks	36	—	3.32	9.97	3.00

Three readings, including the one that concedes ground. The paraphrase gap's magnitude is concentrated in the three tasks with pools under 100: VOC2007 (pool 20, 24.2) and N24News (24, 13.7) give the small stratum an 18.94 mean, and HatefulMemes (pool 2) 6.22 — so a reader who suspects the headline 3.32 is inflated by low-cardinality classification tasks is partly right about the *magnitude*. The ordering claim behaves differently: the 26 large-pool tasks alone reproduce the swap/paraphrase ratio at 3.07 against the pooled 3.00, and the small-pool stratum is where that ratio is *lowest* (1.30), so dropping the small pools sharpens rather than removes the diagnostic. The medium stratum's near-zero mean (0.62) is an averaging artifact, not inertness: it contains both ImageNet-R (15.5) and SUN397 (11.1) alongside ObjectNet (-15.1) and ScienceQA (-4.6), and its swap gap (11.10) exceeds the large stratum's (8.40).

Absolute query counts put the per-task gaps on a concrete scale, recoverable to within one query because each accuracy is a percentage over exactly 300: averaged over the three paraphrases, VOC2007's gap is a *net* loss of 73 of 300 queries, CIRR's 62, ImageNet-1K's 50, ImageNet-R's 46, N24News's 41, SUN397's 33 (net, since a paraphrase also flips some queries from wrong to right; the per-query records in Appendix B.4 count both directions). These are re-aggregations of the accuracies already in `runs/perturb/`, not a new measurement; the paired per-query intervals for this pool-strata split are in Appendix B.4, which re-ran the grid with per-query predictions retained.

Task-normalized effect sizes, and what we recommend reporting. A macro mean of raw point gaps treats one point on HatefulMemes (2 candidates, chance 50) as one point on ImageNet-1K (1,000 candidates, chance 0.1). Dividing each task's gap by its headroom above chance, $\text{orig} - 100/|\text{pool}|$, gives a scale-free version: a paraphrase removes 6.1% of VLM2Vec-2B's headroom on average (median 2.4%), the wrong-task instruction 21.1% (median 11.7%) and deletion 14.8%; swap $>$ para on 31 of 36 tasks, exactly as on the raw scale, and 22 of 26 on the 1,000-candidate tasks alone (5.2% against 15.3%). The audit does not define a threshold of acceptable paraphrase sensitivity, and we do not think one number can: a benchmark that reports a single fixed-wording score should report, alongside it, (i) the mean and worst-of-K paraphrase score over a released paraphrase set, (ii) the headroom-normalized gap per task, (iii) the wrong-instruction score as the ordering reference, and (iv) paired intervals over queries and over tasks. The released audit prints all four; comparability across papers then rests on sharing the paraphrase set, not on a fixed number of them.

B.12 Held-out paraphrase generalization

Every task has a second set of three author-curated paraphrases (`HELDOUT_PARA` in `code/instructions.py`), disjoint from the training paraphrase table. Eight tasks received theirs in a first round (ImageNet-1K, ImageNet-R, VOC2007, SUN397, N24News, CIRR, VisDial and DocVQA); the other 28 were added for the full-budget evaluation at the end of this section. Tasks that share a training paraphrase set share a held-out set, with one exception: DocVQA shares the VQA training set but keeps its own first-round held-out set. No held-out string occurs anywhere in the training table, including across task groups that share an original instruction (ImageNet-1K/A/R share one training set and one held-out set, and the two are disjoint). The held-out strings are never sampled by `instr_aug` and never scored during model selection; they were written under the same constraints as the trained-on paraphrases (Appendix B.3) and kept out of the training of every arm except one: the reassignment arm (Appendix B.29) trains on each string group's first held-out paraphrase as

its alternative wording. They are held out for the A/B/C arms and the task-token arms, not for that arm. Table 20 scores them on the three 500-step pilot arms (seed 0; batch 32, λ=1.0 in the consistency arm), which are distinct from the 800-step pilot (Table 31) and the 2000-step arms (Table 22); training and evaluation use the same split hygiene as Section 6 (`code/perturb_eval.py --conds orig,para1-3,hpara1-3`).

Table 20: Trained-on vs. held-out paraphrase gap for all three 500-step arms (seed 0; the consistency arm uses λ=1.0 at this budget), 8 tasks with disjoint author-curated held-out paraphrases. Lower is better; *tr* is the mean gap over the three trained-on paraphrases, *ho* over the three held-out ones.

	A_base		+instr_aug		+instr_aug+inv_weight	
Task	tr	ho	tr	ho	tr	ho
ImageNet-1K	4.33	6.55	3.44	8.44	0.11	2.11
ImageNet-R	9.12	7.22	1.55	2.78	2.55	3.11
VOC2007	6.45	8.45	5.67	5.77	−0.22	−0.22
SUN397	0.66	6.66	−0.33	4.67	−0.89	2.55
N24News	7.45	7.22	0.77	1.66	0.89	2.11
CIRR	−2.23	−4.67	1.55	2.00	0.56	0.67
VisDial	1.00	3.33	−0.45	2.11	−0.88	1.00
DocVQA	4.00	5.45	1.11	2.11	−1.56	0.67
mean	3.85	5.03	1.66	3.69	**0.07**	**1.50**

Two readings, and both matter. Absolute effect sizes shrink: every arm's held-out gap exceeds its own trained-on gap (3.85 to 5.03, 1.66 to 3.69, 0.07 to 1.50), so the in-distribution numbers in the main text are not the generalized effect size. But the *ordering across arms is preserved* on wordings no arm ever saw — 5.03 untreated, 3.69 with `instr_aug`, 1.50 with `instr_aug` plus `inv_weight` — which is the comparison that matters for whether the fix is real or an artifact of scoring the strings it was trained on. Matched held-out against held-out, `instr_aug` removes 27% of the untreated gap and adding `inv_weight` removes 70%; the corresponding in-distribution figures are 57% and 98%. Read as closure ratios rather than percentages, the full intervention closes the gap $3.4\times$ ($5.03 \to 1.50$) out of distribution against $55\times$ ($3.85 \to 0.07$) in distribution; the full-budget, 36-task version of this comparison is Table 21. Mean *orig* accuracy on these 8 tasks is 41.13 / 40.54 / 42.71, so the held-out reduction is not bought with base accuracy. Per-task, the held-out gap exceeds the trained-on gap on all 8 tasks for `instr_aug` (VOC2007 only marginally, 5.67 to 5.77) and on 7 of 8 for `instr_aug + inv_weight` (VOC2007 exactly flat), but on only 5 of 8 for untreated `A_base` — consistent with the treated arms having fit the specific trained wordings somewhat, which is what a held-out set is for. CIRR's gaps are negative for `A_base` on both sets: the one task where paraphrasing helps the untreated model.

Table 21: Held-out paraphrases on all 36 tasks for the seed-0 2000-step arms (`runs/heldout36_*`, `code/heldout36_arm.sh`). Gaps are macro-mean points below *orig* over 36 tasks; *worst* is the mean per-task worst-of-3 held-out gap; the last column counts tasks whose mean held-out gap is positive.

Arm	orig	hpara_gap	worst	swap_gap	none_gap	hpara>0
A_base	53.43	3.33	6.15	9.18	4.64	29/36
B_aug	53.93	1.56	4.32	7.59	3.53	27/36
C_auginv	53.25	1.08	2.87	9.80	2.13	26/36

All 36 tasks, full budget. With held-out sets for every task, we scored the seed-0 2000-step arms of Table 22 on all 36 tasks under *orig*, hpara1–3, *swap* and *none*, building each arm's candidate cache for the 28 tasks it had not been evaluated on (Table 21). The untreated arm loses 3.33 points to a held-out rewording (worst-of-3: 6.15); augmentation cuts that to 1.56 (4.32) and adding the consistency term to 1.08 (2.87) — $2.1\times$ and $3.1\times$ closures on wordings no arm ever saw — while *orig* stays within 0.7 points and the swap gap within 7.6–9.8 across arms. The held-out gap remains positive on 29, 27 and 26 of 36 tasks respectively, so the residual is broad rather than carried by a few

tasks. The 51 new held-out strings (84 task slots; ImageNet-A, ObjectNet and Place365 reuse their groups' first-round sets, and the nine VQA tasks other than DocVQA share one new set) were drafted with LLM assistance and screened by the authors under the same conventions as the first eight tasks' sets (AI use statement).

B.13 Multi-seed variance for the 500- and 2000-step arms

Table 22: 2000 steps, 8-task subset, mean of three seeds; gaps are points below *orig*. `B_aug` adds `instr_aug`, `C_auginv` also `inv_weight` (exploratory). **Held-out para**: paraphrases never sampled in training; both augmented arms are below `A_base` on it at every seed (Table 24).

Arm	**held-out para gap**	para gap	swap gap	none gap	orig
`A_base`	**8.18**	7.31	15.6	9.28	62.3
`B_aug`	**3.55**	0.24	15.2	6.72	61.8
`C_auginv`	**3.14**	0.45	15.3	6.68	60.3

Table 20 reports the three 500-step arms (`A_base`, `B_aug`, `C_auginv`) at a single training seed. We reran all three at two further seeds (1, 2; disjoint LoRA init and data shuffling only, all other hyperparameters unchanged, including λ=1.0) on the same 8-task subset, 300 queries/task, `code/perturb_eval.py`.

Table 23: Per-seed trained-on para_gap and swap_gap for the three 500-step arms, mean over the same 8 tasks as Table 20.

	para_gap			swap_gap		
Arm	seed 0	seed 1	seed 2	seed 0	seed 1	seed 2
`A_base`	3.85	2.37	2.89	11.92	7.83	7.21
`B_aug`	1.66	2.26	−0.60	6.62	8.21	9.46
`C_auginv`	0.07	0.51	0.19	6.54	8.38	7.83

`B_aug` beats `A_base` on para_gap at every seed (mean closure $3.04 \rightarrow 1.11$, 63%), so that half of the claim survives seed variance. `C_auginv` beats `B_aug` at seeds 0 and 1 (0.07 vs. 1.66; 0.51 vs. 2.26) but not at seed 2, where `B_aug` alone reaches a negative para_gap (-0.60, i.e. paraphrases score *above orig* on average, by fewer than two of 300 queries per task) and `C_auginv` does not (0.19). swap_gap shows no consistent ordering across seeds for any arm pair; all three arms' swap_gaps overlap heavily (6.5–9.5) relative to their spread within a single arm across seeds. We read this as: the `instr_aug` effect on para_gap is robust to seed choice, the incremental effect of `inv_weight` on top of it is not distinguishable from seed noise at this budget and λ, and neither intervention's effect on swap_gap is seed-stable enough to rank the arms on it.

Full budget, three seeds. The 2000-step arms of Table 22 were rerun at seeds 1 and 2 with the same recipe (Table 24). `instr_aug`'s closure of the trained-on gap holds at every seed (7.72, 8.11, 6.11 to -0.04, 0.36, 0.40) and so does its held-out closure (9.43, 8.35, 6.76 to 3.32, 3.97, 3.35). The two effects attributed to `inv_weight` at seed 0 do not reproduce: *orig* recovery (B−A -3.37 at seed 0, but $+1.71$ and $+0.42$ at seeds 1–2; C−B $+3.29$, then -2.37 and -5.63) and the enlarged swap gap (23.42, then 12.04 and 10.54, against 12.75–18.25 for the untreated arm). The consistency term's only seed-stable effect is a 0.2–0.5 point lower held-out gap than augmentation alone, at a cost in *orig* at two of three seeds.

B.14 Fixedness without words: synthetic task-token arms

Section 6's augmentation arm removes the constancy of each task's instruction but also widens the wording distribution the model sees, so it cannot separate three accounts of the paraphrase cost: a memorized *identifier* (the fixed string is used as a task label), *lexical coverage* (the model has simply seen more wordings), and training under *mismatched* query and candidate templates. The arms below vary fixedness with words removed from the equation. Each of MMEB's 23 distinct instruction strings

Table 24: Per-seed values behind Table 22: 2000-step arms, 8-task subset, seeds 0–2. *para* = trained-on paraphrase gap, *hpara* = held-out paraphrase gap (mean of 3 / worst of 3). Seed-0 hpara values come from the 36-task held-out run of the same checkpoints restricted to these 8 tasks; seeds 1–2 were scored with all nine conditions in one run (`code/abl2k_seed.sh`, `code/abl2k_pool_eval.sh`).

Arm	seed	orig	para (worst)	hpara (worst)	swap_gap	none_gap
A_base	0	62.54	7.72 (9.88)	9.43 (13.87)	18.25	9.46
	1	60.46	8.11 (11.71)	8.35 (12.33)	12.75	8.46
	2	63.79	6.11 (9.79)	6.76 (11.04)	15.92	9.91
B_aug	0	59.17	−0.04 (1.46)	3.32 (8.59)	14.92	5.58
	1	62.17	0.36 (1.75)	3.97 (9.42)	16.04	5.62
	2	64.21	0.40 (1.58)	3.35 (6.54)	14.54	8.96
C_auginv	0	62.46	0.26 (1.58)	2.83 (6.29)	23.42	4.79
	1	59.79	0.72 (1.83)	3.43 (6.63)	12.04	8.38
	2	58.58	0.36 (1.21)	3.16 (6.50)	10.54	6.88

is assigned a four-consonant nonce derived from a hash of the string (`instructions.TASKTOK`; tasks that share a string share a nonce, exactly as they share the string), and three LoRA arms are trained with the recipe, budget, data, subset and seeds of Table 6:

- `D_tok` — the query instruction is *replaced* by the nonce, so this fine-tuning never shows the model a natural-language query instruction (candidate templates keep their words, and the base model, Qwen2-VL-2B-Instruct, was instruction-tuned before). Under MMEB's protocol nothing scored distinguishes this model from one that reads its instructions, so the task-ID reading predicts accuracy at `A_base`'s *orig* level when the nonce is supplied (*tok*), and a wrong task's nonce (*tokswap*) costing about what a wrong instruction costs the untreated arm.
- `E_fixtok` — the fixed instruction with the nonce prefixed: two constant identifiers per task. If the model uses whichever fixed cue is cheapest, the paraphrase gap with the nonce kept (*tokpre_para*) should shrink relative to `A_base`, and removing the nonce should cost.
- `F_augtok` — paraphrase augmentation with a *constant* nonce: the identifier is fixed, the words are not. The lexical-coverage account predicts the held-out paraphrase gap of `B_aug` (3.55); the task-ID account predicts it collapses toward zero, because the constant nonce carries the identity the varying words no longer can, and predicts that the right words with the wrong nonce (*tokpre_tokswap*) cost about what *swap* costs.

Every arm is scored under 23 conditions on the 8-task subset: the natural-language grid without any nonce (*orig*, para1–3, hpara1–3, *swap*, *none*, *minedit*), the nonce alone (*tok*, *tokswap*) and the nonce kept with the instruction under each condition (*tokpre_**, including *tokpre_tokswap*). The reference arms are `A_base` and `B_aug` at the same three seeds (Table 6; *minedit* was not scored for them). Tables 25 and 26; code `train_ume.py --task_token`, `token_arm.sh`, `token_summary.py`; results `runs/tok_s{0,1,2}_*_perturb`.

Under this protocol a nonce scores nearly what the instruction does (prediction 1 confirmed). `D_tok`, whose fine-tuning replaced every natural-language query instruction, scores 60.5 ± 2.9 when given only its four-letter nonce, against `A_base`'s 62.3 ± 1.7 with the benchmark's own strings and `B_aug`'s 61.8. This is a stress test of the protocol, not a measurement of how `A_base` or the audited checkpoints use their strings: nothing MMEB scores separates a model that reads a task label from one that reads the instruction. Given the real instructions, which its fine-tuning replaced, `D_tok` scores 54.2, six points *below* its nonce, with a paraphrase gap of 0.33 and a held-out gap of 0.82 (`A_base` 7.31 / 8.18, `B_aug` 0.24 / 3.55), yet a wrong instruction still costs it 10.75 points and a one-word meaning change 6.50. In a model fine-tuned on no query instruction string, then, rewordings are nearly free and wrong instructions are not, the pattern the unseen-string tasks show in the audit (Appendix B.28). That fits a paraphrase cost tied to exposure to the exact string, and an encoder that responds to instruction meaning without instruction training. It does not isolate either: `D_tok` differs from `A_base` in more than exposure, since it was also trained to read a nonce where the instruction goes.

Table 25: Synthetic task-token arms scored with *words only* (no nonce at evaluation), 8-task subset, 300 queries/task, mean over seeds ± half the seed range. Gaps are points below *orig*. D_tok never saw a natural-language instruction in training, so its word-only rows measure what the words are worth to a model trained without them. A_base: fixed instruction; B_aug: paraphrase-augmented; D_tok: nonce replaces the instruction; E_fixtok: fixed instruction + nonce; F_augtok: augmented instruction + constant nonce.

Arm	orig	para_gap	hpara_gap	swap_gap	none_gap	minedit_gap
A_base	62.3 ±1.7	7.31 ±1.00	8.18 ±1.33	15.64 ±2.75	9.28 ±0.73	—
B_aug	61.8 ±2.5	0.24 ±0.22	3.55 ±0.33	15.17 ±0.75	6.72 ±1.69	—
D_tok	54.2 ±3.2	0.33 ±1.49	0.82 ±0.88	10.75 ±1.38	−0.07 ±2.77	6.50 ±1.06
E_fixtok	59.8 ±2.1	5.95 ±1.44	7.11 ±0.66	14.90 ±2.83	9.49 ±2.77	6.78 ±1.25
F_augtok	58.3 ±2.6	0.94 ±1.07	3.55 ±1.74	15.93 ±4.81	5.28 ±1.73	8.10 ±1.31

Table 26: The same arms scored *with the nonce*: *tok* gives only the task's nonce, followed by the separator that ended the instruction, *tokswap* the wrong task's nonce in the same way, *tokpre_orig* the nonce and a space before the fixed instruction; Δ columns are gaps below *tokpre_orig* when the instruction is paraphrased (trained-on / held-out), swapped or deleted, or when the words are right and the nonce is the wrong task's (*tokswap*). With the instruction deleted (Δnone) the input is the nonce, a single space and the query content, which differs from *tok* only in that separator. A_base: fixed instruction; B_aug: paraphrase-augmented; D_tok: nonce replaces the instruction; E_fixtok: fixed instruction + nonce; F_augtok: augmented instruction + constant nonce.

Arm	tok (acc)	tokswap_gap	tokpre_orig (acc)
A_base	—	—	—
B_aug	—	—	—
D_tok	60.5 ±2.9	7.46 ±2.81	56.8 ±1.5
E_fixtok	51.5 ±3.1	4.28 ±0.40	61.2 ±2.4
F_augtok	56.8 ±4.7	2.36 ±1.89	61.1 ±1.2

Arm	Δpara	Δhpara	Δswap	Δnone	Δtokswap
A_base	—	—	—	—	—
B_aug	—	—	—	—	—
D_tok	0.86 ±0.60	2.15 ±1.41	9.82 ±1.73	−2.44 ±1.69	2.71 ±0.19
E_fixtok	5.92 ±0.99	7.83 ±0.95	12.31 ±1.52	16.55 ±2.65	4.01 ±1.19
F_augtok	0.50 ±0.54	2.50 ±0.66	11.83 ±5.06	8.28 ±2.48	4.58 ±1.50

A wrong nonce costs half of what a wrong instruction costs (prediction 2 not confirmed in magnitude). Swapping D_tok's nonce for another task's costs 7.46 ± 2.81, against 15.64 for swapping A_base's instruction; for E_fixtok and F_augtok the wrong nonce alone costs 4.28 and 2.36. The 8-point difference between a wrong nonce and a wrong instruction is of the same size as what a wrong instruction costs D_tok (10.75), but these are accuracies of separately trained models, which differ in what they were trained on as well as in what they are given at evaluation. One model's swap cost cannot be split into components measured on others, so we do not decompose A_base's swap gap into an identity part and a semantic part.

A constant nonce does not absorb task identity when the words vary (prediction 3 not confirmed in its strong form). Giving F_augtok its constant nonce lowers the held-out paraphrase gap from B_aug's 3.55 to 2.50 ± 0.66, not to zero, and the trained-on gap stays at 0.50; the right words with the wrong nonce cost 4.58 — far from *swap*'s 15.9 and below what the wrong nonce costs D_tok. In E_fixtok the nonce kept beside the fixed instruction trims the paraphrase gap from 7.31 to 5.92. Removing the nonce costs 1.4 points (*tokpre_orig* 61.2 to *orig* 59.8). Removing the words costs 9.7 when the nonce is followed by the instruction's own separator (*tok*, 51.5) and 16.55 when a single space follows it (*tokpre_none*, 44.7); the two inputs differ only in that separator, so this arm is also sensitive to the format around its nonce. Models trained with words use the words. The constant identifier is used when present — removing it costs F_augtok 2.8 points and E_fixtok 1.4 — but

it does not replace the words, and adding it removes under a third of B_aug's residual held-out gap (3.55 to 2.50). One graded pattern is worth recording: with words only, F_augtok pays 0.94 for a rewording, 8.10 for a one-word meaning change and 15.93 for the wrong task, the ordering an instruction-follower should show, whereas A_base pays 7.31 for the rewording alone.

What this changes. The arms establish a property of the protocol, not a mechanism of the audited models. Under MMEB's scoring, a model that reads nothing but a per-task nonce comes within two points of one trained on the instruction (60.5 against 62.3), so a fixed per-task string *can* serve as a task-ID token without any score revealing it; this is the sense of the title. The arms do not show how A_base or the checkpoints of Section 3 use their strings, and the stronger reading, that trained models substitute an identifier for the semantics, is not supported where the arms can test it: a wrong nonce costs about half of a wrong instruction, and a constant nonce beside varying words is used but does not replace them. Sections 1, 4 and 6 state the account in the first form only. Caveats: the 8-task subset (Appendix B.20), 300 queries per task, LoRA arms at 2000 steps, three seeds reported as mean $\pm$ half the seed range.

B.15 Query-only vs. candidate-only augmentation ablation

instr_aug as used everywhere else in this paper augments both the query's and the candidate's instruction, drawn independently (Section 6). code/train_ume.py exposes an aug_sides flag (both/qry/pos) that restricts augmentation to one side; we trained two further 500-step arms (batch 32, 25 warmup steps, no consistency term) with it set to qry and pos respectively, one seed each, same 8-task subset as Table 23; the untreated and joint rows are the seed-0, 500-step arms of Table 20.

Table 27: Query-only and candidate-only augmentation against the untreated and jointly-augmented arms, 500 steps, same 8 tasks, one seed each.

Arm	orig	para_gap	swap_gap
A_base (untreated)	41.13	3.85	11.92
D_aug_qry (query-only)	42.25	1.36	9.38
E_aug_pos (candidate-only)	39.00	3.10	7.42
B_aug (joint)	40.54	1.66	6.62

The two gaps dissociate by side. On para_gap, query-only augmentation (1.36) nearly matches joint augmentation (1.66), and both sit well below untreated (3.85) and candidate-only (3.10); at this budget, augmenting the query accounts for essentially all of the paraphrase-gap closure. On swap_gap the ranking flips: candidate-only (7.42) lands nearer joint (6.62) than query-only does (9.38, nearer untreated's 11.92). A lower swap gap is not an improvement. It means the arm separates a genuinely wrong instruction less, which is the sensitivity an instruction-follower should keep, so at 500 steps candidate-side augmentation costs some of what the protocol wants preserved; at 2000 steps joint augmentation leaves the mean swap gap near its untreated level (Table 22). With one seed per arm, we report this as an initial decomposition of *which side* moves *which gap*, not a ratio between them.

B.16 Training setup and hyperparameters for the fix

We fine-tune LoRA adapters (r=16, α=64, dropout 0.1, on q,k,v,o,down_proj, Gaussian init) from the same base checkpoint, varying only instr_aug and inv_weight: AdamW, lr 5×10^{-5}, linear warmup then linear decay, gradient clipping 1.0, GradCache so the effective batch is not limited by activation memory, at most 20,000 samples per MMEB-train source, sources fully mixed (interleave_size 1), bfloat16, one NPU per arm. Training reads MMEB-train and evaluation reads MMEB-eval, which are disjoint splits, so no evaluation query is a training example. Batch size, warmup and λ = inv_weight differ by budget: the 500-step arms use batch 32, 25 warmup steps and λ=1.0 in C_auginv; the 800-step pilot batch 48, 50 warmup steps and λ=0.1; the 2000-step arms, including the task-token arms of Appendix B.14, batch 64, 100 warmup steps and λ=0.1 where the term is on. Neither λ was swept or tuned on an evaluation task. Seeds differ by run, and every table states its own: the 2000-step arms on the 8-task subset (Tables 22 and 24) and the 500-step arms' trained-on and swap gaps (Table 23) are three seeds (0, 1, 2); the 800-step pilot (Table 31), the 500-step held-out panel (Table 20), the 36-task held-out panel (Table 21) and the one-sided

augmentation arms (Table 27) are seed 0 only, so arm differences there carry no seed-level variance estimate.

Three details of the consistency term (equation 3) as implemented (`code/train_ume.py`), since they change what the term means. Gradients propagate through *both* query views: under GradCache, all three encoded views (query variant k_i, positive candidate, query variant m_i) are grad-enabled leaves in the first pass and each receives its cached gradient in the second, so neither view is detached and the penalty is symmetric — it pulls the two encodings together rather than pulling one toward a frozen target. The two query views share the query image and the query's own content byte-for-byte; the instruction-variant index is the only difference between them, with m_i drawn uniformly from the three indices other than k_i, so $m_i \neq k_i$ by construction and no other stochastic transformation is applied to either view. And the candidate side is augmented but not regularized: it draws its variant index independently of the query's, so `instr_aug` does break candidate-template constancy during training, but no term forces two candidate wordings to agree. Omitting candidate-side invariance was a design choice we did not validate — the candidate encoder is itself template-sensitive (Table 33), so a candidate-side or two-sided invariance term is a reasonable variant. One consequence is that the main arms move query-side and candidate-side augmentation together; only the one-seed, 500-step one-sided arms of Appendix B.15 separate the two.

B.17 Training-free paraphrase ensembling and the dispersion gate

Both training-free methods referenced in Section 4, in full.

If the per-task paraphrase effects in Section 3 were per-query noise around a stable, instruction-free signal, a zero-training ensemble — averaging query embeddings under orig plus three paraphrases, holding image and candidates fixed — should denoise that variance and help on average. On an 8-task pilot spanning the brittle/inverted/inert range (n=100/task) we find the opposite in 8 of 8 tasks: ensembling always moves accuracy further in the direction of the task's pre-existing single-paraphrase effect, hurting already-brittle tasks (VOC2007: 73.0 to 55.0; CIRR: 41.0 to 24.0) and helping already-inverted ones nearly to a hindsight ceiling (ObjectNet: 41.0 to 56.0, ceiling 57.0). A mean excluding the original instruction moves further still in the same direction, consistent with the original acting as a partial anchor rather than an interchangeable sample of one underlying signal. That is the signature of a systematic shift in the query embedding; genuine per-query noise would not preserve its sign this cleanly across every task tested.

A natural follow-on is whether an unsupervised, per-query, label-free signal could rescue ensembling by deciding query by query whether to trust it. The most direct candidate — inter-condition embedding dispersion, the mean pairwise cosine distance among a query's four condition embeddings — does not transfer: pooled across the same 8 tasks it predicts which flipped queries ensembling would hurt only weakly (AUC 0.624), and the *sign* of the relationship disagrees across tasks (positive for CIRR/DocVQA/ImageNet-1K/VOC2007/GQA, inverted for OK-VQA/ObjectNet/SUN397), so a threshold tuned on one task's sign would misfire on roughly half the others. On this evidence (one ensemble and one gate, 8 tasks, 100 queries each) the training-free route fails as tried here, though other ensembles or gates may not; the result is also consistent with each task's brittleness or inversion being tied to that task's specific template-content interaction rather than to a general, label-free, geometrically visible property.

B.18 Falsified method hypotheses

Five method hypotheses were falsified by controlled pilots before we arrived at the account in Section 4. We report them as negative results rather than omit them. (i) Cross-task interference is small once pool size is matched (0–2 points on 7 of 10 tasks), so "universal embedders are task-local" as a broad claim is not supported by our data. (ii) A perfect target-type oracle is worth $+4.75$ points, but the best learned type-enforcement probe reaches only $+0.14$ — 3% of the available gain — at any strength, with all confidence gates degenerating to a no-op. (iii) For this checkpoint and a linear probe, target type is not computed-then-discarded internally: leave-one-task-out type decodability is 71–80% at layers 0–27 and peaks at the final layer (82.3%; leave-one-string-out on supported tasks, 63–76% and then 78.3%; Appendix B.30), so we find no sign here of the layerwise mechanism of Li et al. (2026b). (iv) Candidate embeddings are not pure template artifacts: same-image/different-template cosine 0.74 exceeds different-image/same-template cosine 0.40, so content dominates on

the candidate side even though query embeddings are strongly task-clustered (Appendix B.10). (v) Training-free paraphrase ensembling and an unsupervised dispersion gate both fail in the direction the task-ID account predicts (Appendix B.17).

B.19 Does a reasoning stage resist this pathology? A generation-head proxy

A sub-line of 2026 work (nine papers in our local corpus: UME-R1 (Lan et al., 2025), TTE, LaME, PLUME (He et al., 2026), MMEmb-R1 (Wang et al., 2026b), Embed-RL, TRACE, RIME, TWN) adds a generative chain-of-thought stage before pooling, with He et al. (2026) instead placing the reasoning in latent space. Training such a checkpoint is infeasible at our compute scale ($\sim$3680 H20-hours per arm). As a cheap proxy we feed the base (untuned) Qwen2-VL-2B-Instruct's own chat/generation head four of the audit's conditions (*orig*, the first paraphrase, *swap* and *none*) and measure how much its generated reasoning changes, as the cosine between VLM2Vec-Qwen2VL-2B text embeddings (no image) of the generated texts (6 tasks, 30 queries/task/condition). On 5 of 6 tasks the generated text shows the signature a genuine meaning-reader should: paraphrase stays closest to orig (mean cosine 0.816) while swap (0.697) and none (0.714) diverge more — the reverse of what pooled embeddings show on some tasks (Section 4's SUN397 row: $\cos(\text{orig}, \text{para}) \approx \cos(\text{orig}, \text{swap})$). The one task that inverts, VOC2007, is independently the single most paraphrase-brittle of all 36 at pooling time (gap 24.2), and its instruction is a single generic template with no per-query text payload, unlike VQA tasks whose question text anchors the answer. Both symptoms fall on the task with the least per-query text, consistent with content sparsity mattering in both readouts; but a base generation head and a fine-tuned embedding model are different systems, so this cannot localize a shared cause or show whether a trained reasoning stage narrows the pathology.

This is emphatically a smoke test, not an audit of a reasoning-based UME: it uses one untuned base backbone, a generation head rather than a trained embedding readout, and n=30 per task. It cannot tell us how UME-R1 or PLUME would behave under the 6-condition protocol; running that protocol on a released reasoning-driven checkpoint remains open (Section 7).

B.20 Extended limitations

The full version of each limitation summarised in Section 7.

Wordings the fix was trained on. The three paraphrases sampled by `instr_aug` are the same three strings scored at evaluation time, and on 11 of the 12 tasks of the early 800-step pilot (Table 31) all three were seen in training — directly, or via a task that shares MMEB's instruction string (Section 6). The exception, Wiki-SS-NQ, had a baseline gap of only 1.11 and cannot demonstrate closure either way. The held-out check covers the three seed-0 2000-step arms on all 36 tasks (Appendix B.12, Table 21) and three seeds on the 8-task subset (Table 24).

The evaluation-side fixedness counterfactual. Two designs would isolate fixedness from instruction content directly: randomly reassigning semantically equivalent templates across tasks with task and data held constant, or training with synthetic task identifiers uncorrelated with instruction content. We ran the second, on the 8-task subset at three seeds (Appendix B.14); it shows that a semantically empty per-task constant supports near-full MMEB accuracy, which bounds what the protocol can detect but does not show that the natural-language checkpoints use their strings that way. We ran a training-side version of the first: a lottery decides which of two equivalent wordings `A_base` trains on, per string group, on all 36 tasks at two seeds (Appendix B.29). Within classification and within retrieval, the wording that was trained is the one that is privileged; under the design's own randomization test this is significant for the two types together and for retrieval, while classification alone, with 10 possible draws, cannot reach significance. The audited checkpoints themselves were trained on MMEB's strings and are not retested this way, and the arm shows that exposure privileges a wording, not how the model uses it. We regard the task-ID reading as the account most consistent with these measurements, not as an isolated mechanism.

Uncertainty and seeds. Per-query records and paired bootstrap intervals exist for every checkpoint's 36-task audit (Appendix B.4) and for all 23 side-decomposition cells (Appendix B.21): the interaction is -1.03 with task-resampled CI $[-2.41, 0.36]$ and the graded alignment coefficient $\alpha = 0.84$, task-resampled $[0.62, 0.91]$, so the joint gap is sub-additive ($P(\alpha < 1) > 0.99$), while full recovery is rejected by ΔJ's own interval, $[1.68, 10.02]$; the recovered share itself, 0.16, has task-resampled

interval $[-0.10, 0.43]$ and is not distinguishable from zero over tasks. Pool-size stratification and query weighting are reported in Appendix B.11; they show the swap/paraphrase ordering surviving on large pools alone while conceding that the gap's magnitude is concentrated in the three tasks with pools under 100. The 12-task and 8-task ablation subsets were chosen after the audit and on its results, before any training arm was run. The 12-task subset holds the two most brittle tasks (VOC2007, CIRR), the two most inverted (ObjectNet, GQA) and two inert ones (DocVQA, Wiki-SS-NQ): nine IND and three OOD datasets, and no grounding task. The 8-task subset keeps six of these, drops the inverted tasks and Wiki-SS-NQ, and adds ImageNet-R and VisDial: seven IND datasets plus ImageNet-R, all with strings MMEB-train contains. The task-token arms of Appendix B.14 reuse the 8-task subset unchanged. These are diagnostic subsets, not pre-registered ones, and the unselected estimate of the fix's effect is the 36-task held-out panel of Table 21, which no selection touched. The 800- and 2000-step arms use λ=0.1, set once by inspection in the 800-step pilot; the 500-step arms use λ=1.0 (Appendix B.16); neither was tuned on an evaluation task. We reran the three 500-step arms at two further seeds (Appendix B.13): `instr_aug`'s para_gap closure over untreated holds at all three seeds, but the incremental effect of `inv_weight` on top of `instr_aug` reverses at one seed and cannot be distinguished from noise at this budget and λ with only three seeds. The 2000-step arms were also run at seeds 1 and 2 (Table 24): the `instr_aug` closure holds at every seed on trained-on and held-out wordings, and the two effects once attributed to `inv_weight` at seed 0 (orig-accuracy recovery, enlarged swap gap) do not reproduce. Two results remain single-seed: the 800-step pilot, and the 36-task held-out panel of Table 21, which scores the seed-0 checkpoints only; its three-seed counterpart is the 8-task held-out column of Table 24. No λ sweep was run, so whether `inv_weight`'s contribution scales smoothly, saturates, or appears only at these values is unknown; the 500- and 2000-step arms also differ in λ, batch size and warmup, so the two budgets do not isolate training length.

Model and benchmark coverage. The most informative unaudited architectures are late-interaction multi-vector retrieval (Xiao et al., 2025), which scores query–candidate token pairs rather than one pooled vector and so may change the alignment behavior of Section 5, and production-scale MMEB-V2 systems (Zhou et al., 2026). We audited a third checkpoint, Qwen3-VL-Embedding-2B — a different chat template and last-token pooling, not a Qwen2-VL derivative — under the same 6-condition protocol on all 36 tasks at 300 queries/task, so its numbers are scope-identical to the other checkpoints. Aggregated: orig 59.74, paraphrase gap 6.40, worst-paraphrase gap 8.30, swap gap 10.98, deleted-instruction gap 8.12, swap/paraphrase ratio 1.7×, with swap costing more than paraphrase on 29 of 36 tasks (p=3.1×10^{-4} sign test) — against 31/36 (p=1.3×10^{-5}) for VLM2Vec-2B, 28/36 (p=1.2×10^{-3}) for VLM2Vec-7B, 25/36 (p=0.014) for Qwen3-VL-Embedding-8B, and 18/36 (p=1.0) for GME (below), Section 3, Table 2. Table 28 gives the per-task gaps. The qualitative signature replicates on a non-Qwen2-VL backbone, but the ratio does not order the way a backbone-level account would predict: Qwen3's 1.7× sits beside VLM2Vec-2B's 3.0×, and it gets there by being roughly twice as paraphrase-brittle as VLM2Vec-2B (6.40 against 3.32) at a comparable swap gap (10.98 against 9.97); GME is placed on the comparison only with its numbers stated in full, because it behaves unlike every other checkpoint. Under GME's own chat-formatted prompt (the MMEB string in the user role under GME's default system prompt, a stated deviation since MMEB fuses instruction and content into one string), all 36 tasks give orig 51.55, paraphrase gap 1.41, worst-paraphrase gap 3.56, swap gap 2.07 and deleted-instruction gap 0.49 (ratio 1.5×), with swap costing more than paraphrase on 18 of 36 tasks (sign p=1.0, Wilcoxon p=0.69) and on 14 of the 26 large-pool tasks. Its query embedding also moves far less under any perturbation (cosine to the original 0.92 / 0.86 / 0.89 for paraphrase / swap / none, against VLM2Vec-2B's 0.78 / 0.67 / 0.65). The reading is not that GME reads meaning better — a semantic reader would show a large swap gap — but that its *mean* response to the instruction is small: deleting it costs half a point on average. The mean is partly cancellation. Per-task changes have mean absolute value 3.9 under deletion and 4.9 under swap (VLM2Vec-2B: 9.2 and 11.6), exceed 5 points on 10 and 11 tasks respectively, and take both signs (ImageNet-R gains 26.0 points under its swap donor, ImageNet-A loses 22.0, OVEN gains 9–11 under every perturbation), so GME is not indifferent to the instruction and the small macro ordering is not carried task-wise. Two earlier GME grids are withdrawn: the first was scored against a candidate cache contaminated with a VLM2Vec LoRA (the 1.6× ratio); the second encoded queries under VLM2Vec's bare-string convention against GME-format candidates, a launch error that collapsed 26 of 36 tasks to near-floor accuracy and produced a "both gaps near zero" reading. The diagnostic that localized the error and the corrected run are `code/gme_promptstyle_diag.sh` and `code/rerun_gme_clean_v2.sh`; `code/gme_final_assemble.sh` reproduces every

number in this paragraph from `runs/perturb_gme_final/`. The ratio is also sensitive to task composition: on an earlier 8-task audit subset (VOC2007, ImageNet-1K, ImageNet-R, SUN397, N24News, CIRR, DocVQA, RefCOCO; not the ablation subset of Section 6, which has VisDial in place of RefCOCO) the ratios are 2.4× for Qwen3 and 1.7× for VLM2Vec-2B — Qwen3-2B and VLM2Vec-2B exchange places relative to their full-benchmark values while GME's subset swap_gap is *negative* (-1.79 against a paraphrase gap of 1.89), carried by ImageNet-R's -26.0. Any ratio quoted on a subset should be read as a property of that subset; every cross-checkpoint ratio in this paper is full-benchmark. The recipes that most plausibly interact with the mechanism — hard-negative and judge-mined training (Gu et al., 2025a;b), task scaling (Wu et al., 2026b), multi-turn contrastive training (Gu et al., 2026), and reasoning-driven and latent-reasoning readout (Lan et al., 2025; He et al., 2026) — remain unaudited, as does any explicitly hard-negative-amplified (Xue et al., 2025) or reasoning-oriented (Wang et al., 2026b) checkpoint, or any benchmark beyond MMEB-V1 and MMEB-V2's visual-document split (Appendix B.6). We cannot rule out that some recipe trains this memorization out incidentally.

Scaling within the Qwen3-VL-Embedding family. We additionally audited the 8B variant of the same checkpoint family under the identical 6-condition, 36-task, 300-query protocol (`runs/qwen3ve8b_perturb`), so the 2B/8B pair is a same-architecture size comparison rather than a cross-backbone one. Aggregated (gap = orig minus condition, as in Table 2): orig 61.99, paraphrase gap 4.04, worst-paraphrase gap 7.24, swap gap 8.53, deleted-instruction gap 1.17, swap/paraphrase ratio 2.1×, with swap costing more than paraphrase on 25 of 36 tasks (sign test p=0.014; Wilcoxon p=0.0019). Table 28 (right-hand columns) gives the per-task gaps. Going from 2B to 8B lowers both gaps (paraphrase $6.40 \rightarrow 4.04$, swap $10.98 \rightarrow 8.53$) and the ratio moves toward VLM2Vec-2B's 3.0× rather than away from it ($1.7\times \rightarrow 2.1\times$), so within this one family, more parameters make the checkpoint less paraphrase-brittle without closing the swap>paraphrase gap — the qualitative signature is stable under a 4× parameter increase even as its magnitude shrinks. This is one family at two sizes, not a scaling law. The one other size pair we audit, VLM2Vec-Qwen2VL-2B/7B, repeats the paraphrase direction ($3.32 \rightarrow 2.53$) but not the swap direction ($9.97 \rightarrow 12.14$).

Paraphrase adequacy. Our own attempt at the cheapest automated version, the local 2B base model as a text-only judge, failed as an instrument (returning a constant score with zero variance even on sanity cases), which is a negative result about that attempt, not a claim about paraphrase quality.

Swap donor calibration. A wrong instruction can differ from the original along many uncontrolled dimensions at once — length, lexical overlap, syntactic form, target-type cues, compatibility with the candidate index. We ran the lexical half of the graded design: four donors per task, three of them picked deterministically at separated lexical distances, all 36 tasks, reported in Appendix B.2. The swap>paraphrase ordering survives every stratum. The one-word *minedit* and the *shuffle* controls of the same appendix (macro gaps 2.99 and 3.66, against 3.32 for a paraphrase) add a first semantic probe: changing the stated task by one word, or scrambling every word, costs about what a rewording costs. What is still missing is donors characterized by *rated* meaning distance rather than token overlap, spanning instructions from nearby to distant tasks, and more than one donor per stratum, without which the per-task spread we measure (mean 15.8 points) cannot be decomposed into donor-property versus task-property variance.

Pool analyses. The curated-pool multiplier comes from a single-draw log-linear interpolation and the 29.0% false-negative share from text-overlap heuristics rather than a semantic judge (Appendices B.8, B.9).

What the released protocol leaves open. The released 6-condition pipeline (Appendix A) scores a checkpoint against whatever paraphrases and swap donors the user supplies; it does not itself supply annotation guidelines for what counts as a valid paraphrase, semantic-equivalence validation beyond our own judgment and one three-rater check (Appendix B.3), multiple independently-authored paraphrase sets per task, swap donors calibrated on a rated semantic-distance scale rather than lexical overlap, or a predeclared aggregation rule (mean vs. worst-paraphrase, which conditions to report) fixed before results are seen. Two different audit authors following only the released code could therefore report different robustness numbers for the same checkpoint. We release the scoring mechanics and a reporting recommendation (Appendix B.11), not a standard.

Table 28: Qwen3-VL-Embedding-2B and -8B per-task gaps under each instruction condition, all 36 MMEB-V1 image tasks, 300 queries/task. Same conventions and task order as Table 32.

	Qwen3-VL-Embedding-2B					Qwen3-VL-Embedding-8B				
Task	orig	Δpara	Δworst	Δswap	Δnone	orig	Δpara	Δworst	Δswap	Δnone
Classification										
ImageNet-1K	74.33	2.33	3.66	13.66	5.66	80.33	0.55	1.00	13.33	3.00
N24News	64.33	6.00	10.33	40.66	13.33	66.33	3.66	7.00	15.66	5.66
HatefulMemes	66.67	13.01	13.34	10.34	9.67	56.33	5.55	7.33	9.66	8.33
VOC2007	83.33	0.66	1.66	1.00	0.66	84.00	0.67	3.00	2.67	8.67
SUN397	77.33	−0.12	1.66	3.00	1.66	80.00	1.56	2.67	1.33	1.00
Place365	49.67	1.45	2.67	9.00	2.67	46.00	−1.89	−1.33	7.67	−3.33
ImageNet-A	59.67	−3.22	−2.00	0.00	13.00	69.67	−1.33	0.34	6.67	3.00
ImageNet-R	89.67	1.78	2.67	2.34	2.67	89.00	0.67	1.33	6.00	2.00
ObjectNet	83.33	1.44	2.00	8.33	5.33	82.33	−0.12	0.33	11.00	3.66
Country211	23.67	1.67	3.00	8.67	9.67	28.33	0.44	1.33	12.66	13.00
VQA										
OK-VQA	38.67	14.89	16.67	14.00	15.67	44.00	14.78	23.33	−1.00	−1.33
A-OKVQA	31.33	15.00	16.33	12.66	14.33	38.67	16.34	23.00	5.00	−2.00
DocVQA	39.00	13.33	14.00	16.33	13.33	44.00	7.00	13.67	19.67	−21.00
InfographicsVQA	11.00	4.33	4.67	7.33	3.33	12.00	2.11	4.33	4.67	−3.00
ChartQA	32.00	22.33	22.67	21.67	23.67	18.67	7.67	9.67	2.34	1.00
Visual7W	32.00	19.67	20.00	22.00	20.33	36.67	22.12	25.34	25.00	1.67
ScienceQA	45.67	14.78	17.00	19.67	13.67	50.00	15.33	21.33	11.33	−6.00
VizWiz	41.33	8.22	9.33	13.66	9.00	36.67	2.89	7.34	5.67	0.34
GQA	67.67	35.11	43.00	39.67	26.00	76.00	34.33	50.00	46.33	−5.00
TextVQA	70.33	26.88	28.33	44.66	29.00	56.33	4.88	22.66	31.00	−15.67
Retrieval										
VisDial	77.33	−0.34	1.66	0.66	0.33	81.33	0.22	2.66	−1.67	−3.00
CIRR	23.33	3.89	5.00	6.33	5.00	18.33	0.66	3.00	−1.34	0.00
VisualNews_t2i	67.67	−0.77	−0.66	−1.00	−1.00	77.33	0.66	2.00	0.66	2.00
VisualNews_i2t	73.00	1.44	2.67	0.67	2.33	83.00	1.45	3.00	4.00	3.00
MSCOCO_t2i	81.33	2.00	4.66	−0.34	2.66	80.67	0.12	1.34	−1.33	1.00
MSCOCO_i2t	71.00	1.56	2.67	9.00	0.33	74.00	1.78	2.67	4.00	−0.33
NIGHTS	67.33	−0.34	0.33	0.66	1.00	65.67	−2.11	−1.66	−2.33	−3.33
WebQA	91.67	1.22	3.67	8.34	5.34	91.33	0.11	1.00	1.66	0.33
FashionIQ	12.67	1.56	3.00	6.67	2.67	14.00	2.33	2.67	0.33	5.33
Wiki-SS-NQ	79.00	0.67	1.33	3.00	4.00	80.33	0.33	1.00	2.00	1.33
OVEN	68.33	−0.78	−0.34	0.66	1.33	71.33	−1.45	−1.00	2.00	0.33
EDIS	85.00	−1.56	−1.00	0.33	0.00	92.00	0.22	1.33	4.00	1.00
Grounding										
MSCOCO	40.33	−2.00	0.33	3.33	2.00	65.33	6.66	10.33	17.00	14.66
RefCOCO	78.00	13.22	26.33	16.33	7.67	76.67	−7.77	−3.66	8.00	3.00
RefCOCO-Matching	74.33	7.55	13.33	17.33	17.00	76.00	2.45	5.67	10.67	7.00
Visual7W-Pointing	79.33	3.66	4.66	14.66	9.00	89.00	2.56	6.67	22.67	15.67

B.21 Candidate-only and joint perturbation: raw accuracies and templates

Backing detail for Section 5. Raw accuracies (%) under each condition, 300 queries per cell, VLM2Vec-Qwen2VL-2B, one seed. *orig* is both sides in MMEB's wording; *query* rewrites the query instruction only (the Section 3 protocol); *cand* rewrites the candidate template only; *joint* rewrites both sides, pairing query paraphrase k with candidate paraphrase k (the two sides have separate paraphrase sets). All three side tables (Tables 5, 29 and 30) are computed from one set of per-query records: *orig* and *query* are the main audit grid's own records (`runs/perturb_pq`), *cand* and *joint* the side grid's (`runs/candside_pq`), and `code/sidesraw_table.py` checks

every row against the cell gaps behind Tables 5 and 30. A first side run without per-query records (`runs/candside`), whose *orig* and *query* cells were encoded separately from the audit grid, is superseded and used nowhere.

Table 29: Raw per-condition accuracy for the side-decomposition grid, all 23 cells, from the same per-query records as Tables 5 and 30 (orig−mean of each column reproduces their ΔQ = 3.33, ΔC = 3.16, ΔJ = 5.46).

Task	para	orig	query	cand	joint
CIRR	1	47.67	25.33	42.67	25.33
CIRR	2	47.67	26.33	43.33	25.67
CIRR	3	47.67	29.67	49.33	32.67
FashionIQ	1	14.00	11.33	13.00	11.67
FashionIQ	2	14.00	10.67	12.00	7.67
MSCOCO	1	63.00	63.33	59.33	61.33
MSCOCO	2	63.00	63.33	57.33	58.33
MSCOCO_t2i	1	77.33	75.67	67.33	65.67
MSCOCO_t2i	2	77.33	74.67	72.00	69.67
MSCOCO_t2i	3	77.33	76.00	73.00	72.33
NIGHTS	1	66.00	65.00	64.00	65.33
NIGHTS	2	66.00	64.67	67.00	67.33
NIGHTS	3	66.00	64.00	62.67	66.33
RefCOCO	1	81.33	80.67	84.00	83.67
RefCOCO	2	81.33	83.00	75.67	78.00
VisDial	1	72.00	72.33	68.00	67.67
VisDial	2	72.00	73.00	59.67	60.67
VisDial	3	72.00	74.33	72.33	71.33
Visual7W-Pointing	1	68.67	70.00	72.67	71.33
Visual7W-Pointing	2	68.67	69.00	66.67	66.67
VisualNews_t2i	1	72.00	70.00	63.33	64.33
VisualNews_t2i	2	72.00	70.67	71.67	71.00
Wiki-SS-NQ	1	52.00	49.33	49.33	49.33
mean, 23 cells		62.57	59.23	59.41	57.10

Which tasks admit the experiment. A candidate-side rewrite needs a candidate-side template to rewrite, and MMEB's 36 tasks fall into three groups. On 10 the candidate text is a constant template alone (CIRR, VisDial, MSCOCO_t2i, VisualNews_t2i, NIGHTS, FashionIQ, Wiki-SS-NQ, MSCOCO, RefCOCO, Visual7W-Pointing), one of three templates paired with the candidate image. On 4 (WebQA, OVEN, EDIS, RefCOCO-Matching) a template prefixes per-candidate text; the experiment is defined there but was not run. On the other 22, the classification and VQA tasks and the two text-target retrieval tasks, the candidate string is bare content — a class name (`"chair"`), a section label (`"Travel"`), an answer span — so there is no template to paraphrase and the condition is undefined. The unit of the grid is a (task, candidate paraphrase) cell, not a task: of the 30 cells the 10 template-only tasks allow, 23 were run. A first pass covered 6 tasks, chosen to span the query-side brittleness range plus the two largest-pool retrieval tasks, and stopped RefCOCO after 2 paraphrases and Wiki-SS-NQ after 1 because each cell re-encodes 8,416 and 10,694 candidates; a second pass added FashionIQ, MSCOCO, Visual7W-Pointing and VisualNews_t2i at 2 paraphrases each. The grid thus covers every template-only task, but not every cell, and none of the 4 prefix-template tasks.

Candidate templates and their paraphrases. The rewrites reuse the same candidate-side paraphrase sets the training-time fix samples from (`code/train_instructions.py`), so the evaluation-side conditions and the training-side augmentation are not two different wording distributions. For the template shared by CIRR, FashionIQ, MSCOCO_t2i, NIGHTS and VisualNews_t2i:

`orig`: "Represent the given image."
`para1`: "Encode this image."
`para2`: "Produce a representation of this picture."
`para3`: "Embed the image shown."

VisDial and Wiki-SS-NQ share the same set without the terminal period; RefCOCO, MSCOCO grounding and Visual7W-Pointing use the "Represent the given cropped image of the object" family. The training-side sets also hold a Wikipedia-image family (WebQA's), which is sampled in training only; no evaluation cell uses it.

Cost, and why the grid stops at 23 cells. Candidate embeddings are cached keyed by literal candidate text (`mmeb_eval.py`'s `cand_id`), so every candidate-side condition invalidates the cache and re-encodes the task's full candidate pool — 1,002 candidates for MSCOCO_t2i, 2,064 for VisDial, 2,197 for CIRR, 8,416 for RefCOCO, 10,694 for Wiki-SS-NQ — at roughly 0.6 candidates/second/die with images at the 1344 resolution setting (Appendix A). Each cell is therefore a fresh encode of a task's pool, not a re-aggregation of existing embeddings. This cost is what stopped the grid at 23 of the 30 template-only cells and kept the 4 prefix-template tasks out of it; every cell that ran is reported.

Per-task gaps. Table 30 gives the same paraphrase gaps as Table 5, broken out by task rather than aggregated over all 23 cells; every cell has paired per-query records.

Table 30: Paraphrase gaps (points below *orig*) by which side is rewritten, per task, averaged over each task's available candidate paraphrases (23 cells, all with per-query records). Same conventions as Table 5; the last row is the 95% interval resampling tasks, then queries within each drawn task's cells (`code/sides_alpha_variants.py`, 10,000 replicates), the resampling scheme every interval in Section 5 uses.

Task	n para	ΔQ	ΔC	ΔJ	ΔQ+ΔC	inter.
CIRR	3	20.56	2.56	19.78	23.11	−3.33
FashionIQ	2	3.00	1.50	4.33	4.50	−0.17
MSCOCO	2	−0.33	4.67	3.17	4.33	−1.17
MSCOCO_t2i	3	1.89	6.56	8.11	8.44	−0.33
NIGHTS	3	1.44	1.44	−0.33	2.89	−3.22
RefCOCO	2	−0.50	1.50	0.50	1.00	−0.50
VisDial	3	−1.22	5.33	5.44	4.11	1.33
Visual7W-Pointing	2	−0.83	−1.00	−0.33	−1.83	1.50
VisualNews_t2i	2	1.67	4.50	4.33	6.17	−1.83
Wiki-SS-NQ	1	2.67	2.67	2.67	5.33	−2.67
all 23 cells	23	**3.33**	**3.16**	**5.46**	**6.49**	**−1.03**
Task-resampled 95% CI		[−0.10, 8.64]	[1.46, 4.81]	[1.68, 10.02]		[−2.41, 0.36]

The interaction term in full. ΔJ$-(\Delta$Q$+\Delta$C$)$ averages -1.03 points over the 23 cells, is negative in 16, exactly zero in 2 and positive in 5; it exceeds 2 points in magnitude in 8 cells, and its per-cell query-resampled interval excludes zero in 2 (CIRR k=1, NIGHTS k=3). The most sub-additive cells are CIRR's (-5.00, -3.67, -1.33) and NIGHTS's (-5.67); on CIRR the query-only condition has already removed most of the recoverable accuracy (rewriting the query alone costs 18–22 points, and rewriting the index to match gives back 0, -0.7 and 3.0 of them across the three paraphrases), so saturation rather than an alignment effect is the natural reading there. On MSCOCO, MSCOCO_t2i, VisDial and VisualNews_t2i the candidate-side gap runs several points above the query-side gap.

Paired intervals over all cells. Every cell in Table 30 was re-run with per-query records (`runs/candside_pq`, `code/sides_interaction_ci.py`, 10,000 replicates), so the interaction and the graded alignment model $\Delta\text{J} = \alpha(\Delta\text{Q}+\Delta\text{C})$ can be tested with a paired bootstrap rather than by comparing macro means. Resampling queries within cells: $I = -1.03$ $[-1.84, -0.22]$, $\alpha = 0.84$ $[0.75, 0.90]$. Resampling tasks, then queries within each drawn task's cells (the scheme every interval in Section 5 and Table 30 uses): $I \in [-2.41, 0.36]$, $\alpha \in [0.62, 0.91]$, $P(I < 0) = 0.92$, $P(\alpha < 1) = 0.998$. Resampling cells rather than tasks (`code/sides_interaction_ci.py`) gives the narrower $I \in [-2.22, 0.13]$, $\alpha \in [0.71, 0.91]$, $P(I < 0) = 0.96$, which lets cells of one task move independently and is not the interval we quote. These statistics answer different questions. $\Delta\text{J} = 5.46$ $[1.68, 10.02]$ rejects full recovery directly. The slope $\alpha < 1$ indicates sub-additivity across cells, but a through-origin fit weights each cell by the square of its ΔQ$+\Delta$C, so CIRR's three cells carry about three quarters of it. The mean interaction I and the recovered share are the task-level quantities, and under task resampling neither is distinguishable from zero. The reading in Section 5

follows: the strong alignment account is rejected, the pattern leans sub-additive, and ten tasks cannot separate a mostly-independent pattern from a strictly additive one. An earlier version of this test on the 4 tasks then holding per-query records (8 cells) gave $I = -0.42$ $[-1.62, +0.79]$ and was underpowered; the 23-cell run supersedes it.

Estimator details and variants. The paper's α is the least-squares slope of ΔJ on ΔQ+ΔC through the origin, one unweighted observation per (task, paraphrase) cell, so tasks with two or three usable candidate paraphrases contribute two or three cells. Intervals resample *tasks* with replacement and then queries within every cell of each drawn task, so a task's cells move together (`code/sides_alpha_variants.py`, 10,000 replicates; `runs/candside_pq/alpha_variants.json`). Variants: with an intercept, $\Delta J = a + b(\Delta Q{+}\Delta C)$ gives $a = 0.00$ $[-1.22, 1.66]$ and $b = 0.84$ $[0.60, 0.92]$, $P(b < 1) = 0.998$; averaging cells within task and fitting over the 10 tasks gives $\alpha = 0.84$ $[0.60, 0.95]$; the recovered share $1 - \Delta J/(\Delta Q{+}\Delta C)$ is 0.16, $[0.04, 0.26]$ under query resampling and $[-0.10, 0.43]$ under task resampling; the direct test of $\Delta J = 0$ gives $\Delta J = 5.46$ $[1.68, 10.02]$. Per task, $\Delta J/(\Delta Q{+}\Delta C)$ is 0.86 (CIRR), 0.96 (FashionIQ), 0.73 (MSCOCO), 0.96 (MSCOCO_t2i), -0.12 (NIGHTS), 0.50 (RefCOCO), 1.32 (VisDial), 0.70 (VisualNews_t2i), 0.50 (Wiki-SS-NQ) and 0.18 for Visual7W-Pointing, where both single-side gaps are negative (-0.83, -1.00; sum -1.83), so the ratio has no recovery reading there. The earlier cell-resampled interval $[0.71, 0.91]$ understated the task-level uncertainty and is withdrawn.

B.22 800-STEP PILOT

The 800-step pilot referenced in Section 6, on the 12-task subset (nine IND, three OOD datasets; one seed). `para_worst_gap` is the mean over tasks of each task's *own worst* paraphrase gap (not the worst single task), matching Table 1's row 3.

Table 31: 800-step pilot (12 tasks, seed 0; batch 48): the combined arm (`instr_aug` with `inv_weight` λ=0.1) vs. baseline; the two flags are separated only in the 2000-step ablation (Table 22).

	orig	para_mean	para_gap	para_worst_gap	swap_gap
baseline	51.08	47.61	3.47	5.14	8.25
+instr_aug+inv_weight	53.00	52.86	**0.14**	1.30	10.33

B.23 PER-TASK PARAPHRASE, SWAP AND NONE GAPS (ALL 36 TASKS)

The full per-task version of Section 3's summary. Δpara is *orig* minus the mean of the three paraphrases; Δpara-worst is *orig* minus the task's worst paraphrase; Δswap and Δnone are the same gap for the wrong-instruction and deleted-instruction conditions. All cells are 300 queries against the task's own official candidate pool. Every Δ is a gap in points: positive means the perturbation hurt, negative that it helped.

Two entries illustrate that the paraphrases are meaning-preserving on inspection rather than accidentally degraded. VOC2007's original instruction, "Identify the object shown in the image," rewritten as "Name the object that appears in this picture," costs 23.7 points (24.2 on average over its three paraphrases), although the two strings ask the identical question. On GQA, MMEB's own chosen instruction is outright worse than every alternative we tested: a paraphrase gains 12.2 points (task mean; 9.0 for the worst of three), a semantically wrong swapped instruction gains 14.7, and deleting the instruction entirely gains 10.7.

B.24 EXTENDED RELATED WORK

Detail deferred from Section 2, grouped by how each line of work differs from the audit: what it varies (the instruction, or another input axis), whether a model is scored on its own output or against a fixed candidate index, and whether it is a checkpoint the audit could be pointed at.

Varying the instruction at evaluation time. Kostiuk & Enevoldsen (2026), the nearest antecedent, vary far more wordings than we do (38–45 prompts per task in the second version of their paper,

Table 32: Per-task gaps under each instruction condition, all 36 MMEB-V1 image tasks, VLM2Vec-Qwen2VL-2B. Left: the paper's grid (each task's first 300 queries). Right: the full-query replication of Appendix B.5 (all 1,000 queries, same candidate cache). Every Δ is a gap, *orig* minus the condition, in points: positive means the condition hurt, negative that it helped.

	300 queries					all 1,000 queries			
Task	orig	Δpara	Δworst	Δswap	Δnone	orig	Δpara	Δswap	Δnone
Classification									
ImageNet-1K	79.00	16.56	24.67	32.00	23.67	77.40	14.63	28.20	19.30
N24News	74.33	13.66	15.33	28.66	24.66	73.50	13.77	30.00	23.80
HatefulMemes	60.67	6.22	8.00	13.67	5.67	60.30	10.33	10.70	10.50
VOC2007	74.00	24.22	25.33	20.67	35.33	73.90	24.93	25.30	38.00
SUN397	73.33	11.11	15.33	32.33	15.66	72.90	9.93	28.50	13.40
Place365	33.00	−1.67	0.00	2.67	3.33	35.10	0.47	2.80	4.00
ImageNet-A	52.33	10.89	15.66	52.33	13.00	50.20	14.73	50.20	10.10
ImageNet-R	80.67	15.45	32.67	30.00	17.00	84.60	15.67	16.30	17.10
ObjectNet	36.33	−15.11	−13.67	−12.00	−13.34	36.10	−16.97	−11.90	−15.70
Country211	21.00	−0.33	1.00	18.67	10.33	22.20	−0.17	20.00	11.30
VQA									
OK-VQA	42.67	1.12	1.34	2.00	5.67	48.00	1.13	5.00	6.30
A-OKVQA	39.33	−0.56	0.66	3.66	1.66	39.80	1.17	3.60	4.80
DocVQA	85.00	0.11	1.67	5.67	8.33	83.30	2.27	6.10	9.40
InfographicsVQA	42.67	3.89	4.34	7.34	6.67	47.80	3.87	7.90	9.10
ChartQA	44.00	7.11	13.00	14.00	16.33	42.30	5.47	13.10	12.30
Visual7W	46.67	1.67	4.34	7.34	8.34	51.20	3.83	9.10	10.00
ScienceQA	32.67	−4.55	−3.00	2.34	0.00	30.70	−3.40	3.30	−0.70
VizWiz	34.00	4.11	8.00	6.33	5.67	38.50	2.90	4.60	5.30
GQA	47.67	−12.22	−9.00	−14.66	−10.66	48.40	−12.93	−16.00	−12.30
TextVQA	65.00	3.00	5.00	6.67	7.00	63.30	2.57	4.80	4.40
Retrieval									
VisDial	72.00	−1.22	−0.33	0.00	−2.00	74.10	1.00	1.70	−0.20
CIRR	47.67	20.56	22.34	20.67	26.34	46.60	20.87	23.30	27.80
VisualNews_t2i	72.00	1.67	2.00	2.33	2.67	73.40	0.23	−0.10	1.80
VisualNews_i2t	74.00	2.22	4.67	13.00	5.33	75.10	3.33	13.10	5.10
MSCOCO_t2i	77.33	1.88	2.66	0.66	4.66	72.40	1.10	2.00	3.70
MSCOCO_i2t	67.67	4.00	4.67	19.34	7.34	68.30	2.43	16.50	7.60
NIGHTS	66.00	1.44	2.00	5.67	0.33	65.80	0.70	4.40	0.80
WebQA	87.67	1.67	4.34	5.34	7.67	86.00	1.77	5.30	5.70
FashionIQ	14.00	1.33	3.33	4.67	4.00	14.30	1.73	3.70	5.00
Wiki-SS-NQ	52.00	1.22	2.67	0.00	2.67	53.40	1.17	0.50	2.60
OVEN	66.67	0.11	1.67	2.00	−2.66	67.50	1.13	1.40	−0.40
EDIS	78.33	0.22	1.66	2.00	2.00	80.40	−0.33	0.40	1.70
Grounding									
MSCOCO	63.00	0.45	2.00	13.00	14.00	66.50	0.93	14.70	14.10
RefCOCO	81.33	−0.34	0.66	3.66	6.33	80.70	0.57	7.10	8.20
RefCOCO-Matching	76.67	0.12	1.34	10.00	11.67	75.30	1.23	10.80	13.30
Visual7W-Pointing	68.67	−0.33	0.67	−3.00	0.67	68.40	0.73	−2.80	0.00

against our 3) over a text-only panel of 6 models and 11 datasets rather than one multimodal benchmark family; both observations precede ours. The *swap* control we add converts a claim about *variance across wordings* into one about *ordering*: a model can be prompt-sensitive and still read meaning, and the swap asks how much more it pays for a wrong instruction than for a correct rewrite. Of the five UME papers in our corpus that mention instruction robustness, only Sánchez Aimar et al. (2025) tests it. Their Appendix D.2 trains VLM2GeoVec-2B with one or with ten templates per task and evaluates on a known and on an unseen, rephrased template. Known-template accuracy is

about equal, while moving to the unseen wording costs 10.05 points after single-template training and 3.53 after ten-template training. This is the direction of our Section 6. It differs in scope: one remote-sensing benchmark, no wrong-instruction control, and no audit of what a fixed-string protocol reports for the single-template model. Their "instruction-free" refers to baseline dual encoders, not to an instruction ablation. Choi (2026) measure the text-only counterpart of Section 6: task-specific fine-tuning under a fixed instruction reduces sensitivity to paraphrased instructions by roughly half to two-thirds at 1.7B–4B but not reliably at 8B, and the effect varies across model families, so the closure our augmentation arm achieves should be expected to depend on task count, wording diversity, scale and format — none of which we vary — and the held-out numbers of Appendix B.12 are the ones to carry across settings. GrIPS (Prasad et al., 2023) searches instruction edits systematically and finds edits that degrade coherence yet raise accuracy; an edit-search over MMEB's strings is the natural next audit beyond three curated paraphrases per task.

The same question on generative models. Navarro Carranza (2025) stress-test LLMs with question paraphrases and read the drops as surface-form reliance, the reading we apply on the embedding side with a swap control; Ismithdeen et al. (2025) audit 61 prompt formulations across 10 large multimodal models on multiple-choice QA and find accuracy swings of up to 15 points from phrasing alone; Epstein et al. (2024) and Shiono et al. (2025) evaluate whether vision-language models follow instructions dispersed across multi-turn dialogue or lost after visual instruction tuning; Yang et al. (2024) vary instruction forms for generative MLLMs across multimodal reasoning tasks and report stability across them, a design adaptable to embedding retrieval and roughly what our released drop-in does. The closest text-only analogue is Razavi et al. (2025): their finding that models trained on one wording generalize poorly to its paraphrases is the pattern we measure on the embedding side. VisIT-Bench (Bitton et al., 2023) evaluates vision-language instruction following on 592 queries whose instructions vary per instance across 70 instruction families, the design under which following can be scored as such; MMEB's one-string-per-task retrieval protocol cannot, which is the gap the audit measures. What these studies can conclude differs from ours because of the candidate pool: a generative benchmark scores a produced string, so a rewrite changes only the input, whereas here it changes a query vector that must still rank against an index encoded under the original convention, which makes query–index alignment a separate failure channel (Section 5).

Other input axes. Musacchio et al. (2026)'s token-format finding is compatible with, and orthogonal to, ours; a format check we ran while reproducing VLM2Vec-Qwen2VL-2B (ImageNet-1K, 150 queries, identical weights and data, before we matched the official image pre-resize of Appendix A) moved accuracy between 56.0 and 71.3 across 18 combinations of image-token placement and candidate-side whitespace, consistent with format as a separate axis; we did not repeat it under the final pipeline. Input resolution is another: Guo et al. (2026) report VLM2Vec-style models at 336 and at 1344 input resolution as a scaling configuration, and Qin et al. (2025) exclude VLM2Vec's 1344 Qwen2-VL variant from comparison with their own 672 setting as a fair-comparison confound. What we newly report (Appendix A) is narrower: the 1344 setting force-resizes every image to a square, discarding aspect ratio, on a backbone with native dynamic-resolution support that does not require this, at a cost of 10–26 points on grounding tasks if dropped without adjustment. This is a supporting reproducibility finding, not the paper's central claim.

How instructions enter training. Asai et al. (2023) train a dense retriever to read a natural-language task description alongside the query, the setting MMEB inherits; the question here is whether the description is read or matched. Xu et al. (2023) pair each of 62 multimodal tasks with five instruction templates during tuning; per-task instruction diversity is thus the norm in generative instruction tuning, the convention MMEB's one-string protocol departs from, and `instr_aug` (Section 6) imports it into embedding training. Yang et al. (2025) select visual-instruction data by worst-case perturbation subgroups, a robustness-aware data-selection strategy distinct from both format mixing (Ngweta et al., 2025) and the paraphrase augmentation used here. On the representation side, Li et al. (2024) argue an instruction's embedding should capture the *task* rather than its surface semantics. That is the legitimate version of what we find MMEB rewarding by accident, and the contrast we draw: a representation encoding task semantics survives paraphrase; a fixed surface template that marks its task (or group of tasks sharing it) only because it never varies does not.

Checkpoints and designs we do not audit. The strongest current recipes include hard-negative and judge-mined training (Gu et al., 2025a;b; Xue et al., 2025), task scaling (Wu et al., 2026b), multi-turn objectives (Gu et al., 2026), varied visual-token budgets (Li et al., 2026a), reasoning-driven readout (Lan et al., 2025; He et al., 2026; Wang et al., 2026b; Cui et al., 2025), mixture-of-experts scaling

(Cui et al., 2026), pair-aware ranking (Chen et al., 2026), late-interaction retrieval (Xiao et al., 2025), further readout and input-allocation designs (Wang et al., 2025; Wu et al., 2026a; Sun et al., 2026; Wang et al., 2026a; Jin et al., 2026) and production-scale MMEB-V2 systems (Zhou et al., 2026). The released protocol's design transfers to each of them, though scoring code may need adapting (e.g. multi-vector scoring for late interaction); none is audited here. Each would test something specific. Gu et al. (2025a) trains on MMEB with textual-discriminative distillation and hard-negative enhanced instruction tuning, Gu et al. (2025b) uses an MLLM as a judge to mine discriminative negatives, and Xue et al. (2025) amplifies hard-negative gradient contributions directly; all three change precisely the candidate-pool structure our exploratory pool analysis argues MMEB leaves under-specified, and stronger discriminative training is a plausible partial mitigation of template dependence that we can neither rule in nor out. Wu et al. (2026b) scales the number of training tasks, which bears directly on the account here: we conjecture that a fixed per-task string is usable as a task identifier only while the task inventory is small and static. Gu et al. (2026) trains with multi-turn contrastive objectives, where the instruction is no longer a single static prefix, and Li et al. (2026a) is a contemporary MLLM-based UME family on MMEB with a different visual-token budget, bearing on whether the effect is architecture-general. The two earlier MLLM-based encoders, Jiang et al. (2024) (prompt-conditioned single-modality training) and Lin et al. (2024) (an instruction-tuned multimodal bi-encoder with an explicit modality-bias analysis), would test whether the effect is tied to the VLM2Vec recipe or to fixed-instruction benchmarks generally. Several designs change where an instruction-like signal enters or is stored, and so bear directly on whether a surface template can act as a task identifier: Cui et al. (2025) generate a reasoning context before embedding, so the instruction enters through generated text rather than a fixed prefix; Cui et al. (2026) scale UMEs with expert routing; Wang et al. (2025) condition a shared backbone on multiple *learnable* prefixes and minimize mutual information across the resulting parallel embeddings; Sun et al. (2026) and Wu et al. (2026a) route all information through a small set of bottleneck or latent-reasoning tokens instead of a vocabulary token's hidden state; Wang et al. (2026a) accumulate retrieval evidence in learnable registers over recurrent depth, a readout our layerwise probe (Section 4) does not cover; and Jin et al. (2026) vary the visual-input budget per frame on MMEB-V2 video tasks, where instruction and input-budget decisions interact. Chen et al. (2026) condition on the query–candidate pair rather than encoding each side independently, so their alignment behavior under Section 5's conditions may differ from the dual-encoder checkpoints audited here. Further afield, Kim et al. (2025) replaces dense-index cosine retrieval with generative universal multimodal search; the fixed-index assumption our audit depends on does not apply there, so it is an adjacent formulation rather than an audit target.

The adjacent benchmarks. MMEB-V2 (Meng et al., 2025) and ViDoRe (Faysse et al., 2025) both still pair each task or query with a single fixed template, so the confound this paper measures is not specific to MMEB-V1 nor obviously fixed by either successor's design. Appendix B.6 audits MMEB-V2's visual-document split directly: the swap $>$ paraphrase ordering holds for all three checkpoints tested, and the checkpoint MMEB-V2 was released with scores within about a point of its official number under every rewording tried.

B.25 Training objective in full

The objective summarised in Section 6. Each training step draws a batch of B query/positive-candidate pairs (q_i, c_i^+) from MMEB-train with sources fully mixed. Both sides are encoded by the same LoRA model, pooled at the last token, and L_2-normalized, giving unit vectors z_i^q, z_i^c. The base objective is InfoNCE over in-batch negatives only (no mined hard negatives), at temperature $\tau = 0.02$:

$$\mathcal{L}_{\text{NCE}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp\big(z_i^{q\top} z_i^c/\tau\big)}{\sum_{j=1}^{B} \exp\big(z_i^{q\top} z_j^c/\tau\big)}. \tag{2}$$

Let $\mathcal{I}_T = \{I_T^{(0)}, \ldots, I_T^{(3)}\}$ be task T's instruction variants, where $I_T^{(0)}$ is MMEB's own wording and $I_T^{(1..3)}$ are the same three author-curated paraphrases used in the audit. With `instr_aug`, every item independently draws $k_i \sim \mathcal{U}\{0, \ldots, 3\}$ for its query and, separately, k_i' for its positive candidate, which has its own four-variant template set — a query and its positive are not forced to share a wording. With `inv_weight` $= \lambda > 0$, each query is encoded a second time under a variant m_i

drawn uniformly from the three indices *other than* k_i, and the two encodings are pulled together:

$$\mathcal{L}_{\text{inv}} = \frac{1}{B}\sum_{i=1}^{B}\Big(1 - z_i^q(k_i)^\top z_i^q(m_i)\Big), \qquad \mathcal{L} = \mathcal{L}_{\text{NCE}} + \lambda\,\mathcal{L}_{\text{inv}}. \tag{3}$$

Because both views are already unit vectors, $\mathcal{L}_{\text{inv}}$ is exactly $1 - \cos(\cdot,\cdot)$ and is applied *after* normalization. Gradients flow through *both* views (neither is detached), so the term pulls the two encodings toward each other rather than toward a fixed target; the views share the query image and content byte-for-byte, differing only in the variant index. It is a query-side term: the candidate side is augmented but never regularized, a design choice we did not validate against a candidate-side or two-sided variant, given that the candidate encoder is itself template-sensitive (Table 33) on a subset of tasks (Section 5). We use λ=0.1 in the 800- and 2000-step arms and λ=1.0 in the 500-step arms, and did not sweep it.

B.26 Positive control: content does steer retrieval

The account in Section 4 documents a fixed instruction *template*, constant across every query in a task, which invites the worry that the model may not condition on query content at all. It does. We check this with a zero-new-inference natural experiment: two MMEB tasks (GQA, InfographicsVQA) happen to reuse the same image across multiple cached rows with different questions and answers. We pair every two of a task's 1,000 eval queries that share an image and have different ground-truth answers, and score each query of a pair as a two-way choice between its own answer and the other query's. A query's embedding prefers its own answer in 84.5% of 444 choices on GQA (222 pairs) and 89.4% of 8,142 on InfographicsVQA (4,071 pairs; `code/compositional_probe.py`). The model is plainly steerable by what is asked; the pathology documented here is specific to the constant task-level template, not a wholesale failure to read text, which sharpens rather than undercuts the audit's central claim.

B.27 The mechanism measurements in detail

Methods for the four measurements listed in Section 4, collected here as a table; the target-type probe has its own specification in Appendix B.30.

Table 33: Measurements consistent with the fixed instruction template functioning as a task-identity signal. None isolates the template from co-varying task content, target type, or pool composition.

Measurement	Value
Task identity share of query / candidate embedding variance	32.2% / 32.3%
Distinct candidate instruction strings per task (RefCOCO, 8,416 candidates)	1
One-character candidate-template edit ("image."→"image"), cosine	0.7397 (noise floor 0.9994)
— accuracy cost, MSCOCO_t2i (72.3 → 68.8)	3.5
Same edit on an already-canonical template (VisDial, control), cosine	0.99936 (acc. 74.1 → 74.5)
SUN397: cos(orig, paraphrase) vs. cos(orig, swap)	0.610 vs. 0.620

The variance decomposition is specified in Appendix B.10. The one-character-edit rows use `mmeb_eval.py`'s `--canon_cand_template` flag, which refuses to run unless the task's candidate text is already a single constant template, so no candidate-specific information can be destroyed by the rewrite. The SUN397 row is the task where the audit's paraphrase and swap displacements are closest: a paraphrase and a wrong instruction move the query embedding by a similar amount (cosine 0.610 and 0.620 to *orig*). Equal displacement does not make the two semantically indistinguishable to the model, since the directions can differ; it shows only that the size of the move does not separate a rewording from a wrong instruction there.

B.28 STRING EXPOSURE VERSUS MMEB'S IND/OOD SPLIT

MMEB's IND/OOD split labels *datasets*: the 20 IND tasks are the MMEB-train sources, the 16 OOD tasks are not. It does not say which evaluation *strings* a checkpoint was trained on, because the 36 tasks share 23 distinct query-instruction bodies. Four groups share one string each: all ten VQA tasks; ImageNet-1K/A/R; VOC2007 and ObjectNet; SUN397 and Place365. MMEB-train's 20 sources carry 15 distinct bodies. A task is *string-seen* if its eval query-instruction body equals the training body of some MMEB-train source (MSCOCO grounding uses the train wording of Section 4's natural experiment, which differs from its eval wording). This gives 27 seen and 9 unseen tasks. Eight OOD datasets reuse a trained string (ImageNet-A/R via ImageNet-1K, ObjectNet via VOC2007, Place365 via SUN397, and ScienceQA, VizWiz, GQA and TextVQA via the shared VQA string), and one IND dataset, MSCOCO grounding, is evaluated under a string it never trained on. Gaps are orig minus condition (positive is a cost), VLM2Vec-2B, 300 queries per task; |para| is the absolute value of each task's mean-of-3 paraphrase gap, averaged over tasks. The p-values are two-sided permutation tests on the difference of group means (100,000 relabelings of the 36 tasks); because tasks that share a string are not independent, we also relabel the distinct strings themselves, exactly over every relabeling (`code/exposure_string_perm.py`). Script: `code/exposure_regroup.py`; output: `runs/exposure_regroup.json`.

Table 34: Audit gaps regrouped by MMEB's dataset split and by exact-string exposure, VLM2Vec-2B, 36 tasks. Seen/unseen: whether the task's eval query string occurs in MMEB-train.

Group	n	|para|	para	swap	none
MMEB IND	20	6.07	5.89	12.37	10.92
MMEB OOD	16	4.44	0.12	6.98	3.56
string seen	27	6.96	4.34	11.41	8.38
string unseen	9	0.49	0.27	5.67	5.45
IND, seen	19	6.36	6.18	12.33	10.75
IND, unseen (MSCOCO)	1	0.45	0.45	13.00	14.00
OOD, seen	8	8.38	−0.01	9.21	2.75
OOD, unseen	8	0.50	0.25	4.75	4.38

What the regrouping shows. Exposure separates the *magnitude* of the paraphrase gap: |para| is 6.96 for seen and 0.49 for unseen tasks (p=0.007; relabeling the 23 strings instead, p=0.016 over all 817,190 relabelings), and no unseen task moves by more than 1.33 points (FashionIQ). Within the OOD split alone, the eight tasks with a trained string move 8.38 points on average against 0.50 for the eight without (p=10^{-4}; p=0.002 over the 495 string-level relabelings). The dataset split separates the *sign*: IND tasks lose 5.89 to a paraphrase and OOD tasks 0.12 (p=0.023), largely because the OOD tasks with a trained string move in both directions (gaps of 10.9 on ImageNet-A, 15.5 on ImageNet-R, 4.1 on VizWiz and 3.0 on TextVQA; −15.1 on ObjectNet, −12.2 on GQA, −4.6 on ScienceQA and −1.7 on Place365), so their signed mean is −0.01 against an absolute mean of 8.38. The swap and none gaps do not separate seen from unseen tasks (p=0.26 and 0.44): the nine unseen-string tasks are inert to a paraphrase yet still lose 5.67 points to a wrong instruction and 5.45 to none (MSCOCO 13.0 and 14.0, Country211 18.7 and 10.3, RefCOCO-Matching 10.0 and 11.7).

What it does not show. String exposure is not assigned independently of anything else. Every VQA string is seen and every grounding string unseen, so within task type the comparison rests on classification (nine seen tasks, |para| 12.77, against Country211, 0.33) and retrieval (eight seen, 4.33, against FashionIQ, Wiki-SS-NQ, OVEN and EDIS, 0.72). Permuting seen/unseen labels only within these two types, the seen tasks still move more (one-sided p=0.0006, 200,000 relabelings), but only five unseen tasks enter that comparison. Exposure is also partly confounded with how much of the query's text the instruction is: eight of the nine unseen tasks carry their own text beside it (an object phrase, a caption, a question), and Appendix B.7 finds paraphrase gaps largest where the instruction is the query's only text. Country211 is the one unseen task whose query text is the instruction alone, and it is inert (0.33) where the nine seen classification tasks average 12.77. The regrouping associates the paraphrase cost with exposure to the exact string; Appendix B.29 assigns the exposure by lottery instead.

A correction. An earlier version of our analysis code listed RefCOCO as IND and VOC2007 as OOD, the reverse of MMEB's split; under that list the IND/OOD paraphrase gaps were 4.66 and 1.65 (p=0.26). Every number in this paper uses the corrected list (`code/mmeb_eval.py`); the old list is kept in `exposure_regroup.py` as `MMEB_IND_OLD` only to reproduce the figure it gave.

B.29 Reassigning strings by lottery

Design. The regrouping above compares strings MMEB happened to train on with strings it did not, so exposure goes with task type. This arm assigns exposure by lottery. The 19 MMEB-train sources with an evaluation counterpart use 14 query strings; tasks that share a string share the draw. One draw (seed 0), stratified by type, reassigned 2 of the 5 classification groups (VOC2007, SUN397), 4 of the 8 retrieval groups (CIRR, VisDial, NIGHTS, MSCOCO_t2i) and, by design, the single VQA group, the only way to vary exposure for that type. Every training source in a reassigned group was trained, for the whole run, on one fixed alternative wording of its query instruction, *rstr*: the group's first held-out paraphrase (Appendix B.12; for the VQA group, the one shared by nine of the ten VQA tasks), so this string is held out for the other training arms but trained on in this one. Candidates and every other choice follow `A_base` (2000 steps, batch 64, no augmentation; Appendix B.16), and MSCOCO grounding keeps its train wording, which already differs from its evaluation wording. Two training seeds (0, 1) share the draw. This gives 9 *kept* tasks (trained on orig: ImageNet-1K/A/R, N24News, HatefulMemes and four retrieval tasks), 18 *reassigned* tasks (trained on *rstr*: VOC2007, ObjectNet, SUN397, Place365, the ten VQA tasks and four retrieval tasks) and 9 *unseen* tasks with no training source. Each of the 36 tasks is scored at 300 queries under eight query conditions: orig, *rstr*, four further paraphrases the recipe never trains on (two from the training table, two held-out), swap and none. The prediction, recorded in `runs/reassign_design.json` before any cell was scored: orig $-$ *rstr* > 0 on kept and < 0 on reassigned tasks, of the order of `A_base`'s 36-task held-out gap (3.33; Section 6); the neutral paraphrases below the trained wording in both groups; VQA nearly inert; and no orig/*rstr* preference on unseen tasks.

Table 35: Reassignment arm by group and task type, `A_base` recipe, mean of two training seeds, 300 queries per task. orig $-$ rstr: MMEB's string minus the alternative wording. trained $-$ neutral: the wording the group was trained on (orig for kept and unseen tasks, rstr for reassigned ones) minus the mean of the four neutral paraphrases. The swap gap is orig minus swap in every row, so on reassigned tasks it is measured from an untrained wording.

Group	Type	n	orig	orig $-$ rstr	trained $-$ neutral	swap gap
kept (trained on *orig*)	CLS	5	58.1	+9.3	+4.7	23.5
	RET	4	64.1	+3.3	+1.2	9.8
	all	9	60.8	+6.6	+3.2	17.4
reassigned (trained on *rstr*)	CLS	4	54.2	−2.2	+4.4	6.3
	VQA	10	42.4	−0.8	+0.1	−0.2
	RET	4	54.0	−1.6	+1.9	1.7
	all	18	47.6	−1.3	+1.4	1.7
unseen (no training source)	CLS	1	24.8	−1.0	−1.3	14.8
	RET	4	48.4	+0.3	+0.4	1.5
	GND	4	61.8	+1.3	+0.0	3.4
	all	9	51.7	+0.6	+0.0	3.8

Results. Table 35 averages the two seeds. The trained wording wins in both groups that have one: orig $-$ *rstr* is $+6.6$ on the kept tasks (positive on 8 of 9) and -1.3 on the reassigned ones (negative on 14 of 18). The sign flips within type: classification $+9.3$ (5 kept tasks) against -2.2 (4 reassigned), retrieval $+3.3$ against -1.6 (4 and 4). The neutral paraphrases sit below the trained wording in both groups ($+3.2$ kept, $+1.4$ reassigned), while MMEB's string, once untrained, sits at their level ($+0.15$ on the reassigned tasks). Pooled over the 27 tasks with a training source, the trained wording's advantage is 3.1 points, the predicted order. VQA leans only weakly toward the trained wording (-0.8; *rstr* ahead on 7 of 10), and the unseen tasks show no preference ($+0.6$; $|\cdot| < 2$ on 6 of 9). The lottery costs little accuracy under the trained wording: kept-task orig is 60.8 against `A_base`'s

60.5 (seed 0), and reassigned-task *rstr* reaches 48.9 against `A_base`'s orig 50.6 on the same 18 tasks. Table 36 gives every task.

Inference follows the assignment. The lottery drew string groups within type, not tasks, so the test enumerates the assignments the design could have produced: $\binom{5}{2}$=10 in classification, $\binom{8}{4}$=70 in retrieval, 700 jointly. The VQA group was reassigned by design rather than drawn, and the unseen tasks have no draw, so neither enters the test. Under each assignment we recompute kept minus reassigned orig $-$ *rstr*, holding every task's outcome fixed (the sharp null that which wording a group trains on changes no outcome; `code/reassign_group_perm.py`). On the seed mean the observed draw is the most extreme of the 700 in the predicted direction: two-sided p=0.006 (one-sided $1/700$) for classification and retrieval together. Retrieval alone, where each group is one task, gives p=0.029, the smallest of its 70 draws. Classification alone gives two-sided p=0.20 and one-sided 0.10: its observed draw is again the most extreme, but with 10 possible draws 0.10 is the floor, so classification alone cannot be significant under this design. Averaging each group's tasks before comparing groups leaves all three values unchanged.

What it does not show. The test rests on one draw over 13 randomized groups, 5 of them in classification. The two seeds share that draw, so they are two measurements of one randomization, not two randomizations. They agree in sign within both types, but seed 1 alone is weaker: two-sided p=0.081 over the 700 draws (seed 0: 0.0014) and 0.20 in retrieval alone. The asymmetry between $+6.6$ and -1.3 is mostly the wording: *rstr* is a harder-than-average paraphrase on the kept tasks (untrained minus neutral -3.5), and the trained-minus-neutral margins ($+3.2$, $+1.4$) are closer, so wording quality moves the score alongside exposure. The arm varies exposure for one recipe and one replacement wording per group, and it does not retest the audited checkpoints, which were trained on MMEB's strings. It shows that training on a wording is what privileges it; it does not show that the model then uses that wording as a task identifier. Scripts: `code/reassign_design.py`, `reassign_arm.sh`, `reassign_eval36.sh`, `reassign_analysis.py`, `reassign_table.py`, `reassign_group_perm.py`; outputs: `runs/reassign_summary.json`, `runs/reassign_group_perm.json`.

B.30 The target-type probe, specified once

Specification. One probe is used for every probe number in this paper except where marked. Input: the last-token hidden state of VLM2Vec-2B's query encoding, L_2-normalized, at each of the 29 hidden states (embedding output plus 28 transformer layers; layer 28 is the embedding MMEB scores), for the first 60 eval queries of each of the 36 tasks under the *orig* instruction (`runs/layer_probe_states.npz`). Label: the task's target type, one of seven classes (LABEL, class-name text, 10 tasks; ANSWER, short-answer text, 10; CAPTION, 2; IMAGE, 6; DOCIMG, a document image, 1; WIKI, an image paired with a text passage, 3; CROP, an image region, 4), so chance is 14.3%. Classifier: ridge regression onto one-hot targets with a bias feature, prediction by argmax, penalty $\lambda d\, I$ with λ=10^{-2} and d=1,537 features (`code/type_enforce.py` writes it as λ times the row count of the $d\times d$ Gram matrix; we keep that value so the numbers reproduce). Aggregation: the mean over held-out tasks of per-task accuracy (every task has 60 queries, so this equals pooled accuracy). Three splits: *within-task* (each task's queries split 30/30, all 36 tasks in the training fold); *leave-one-task-out* (LOTO: fit on 35 tasks, test on the held-out one); and *leave-one-string-out* (LOSO: hold out every task that shares the held-out task's query string, so the probe has never seen that string). A held-out task is *supported* if some training-fold task has its type: 35 tasks under LOTO (not Wiki-SS-NQ, the only DOCIMG task) and 25 under LOSO (not Wiki-SS-NQ, and not the ten VQA tasks, since the only ANSWER tasks share one string).

Results. Within a task the probe is near-perfect (96.3%), as a constant string makes trivial; its misses are concentrated in the WIKI and DOCIMG tasks (Wiki-SS-NQ 50.0%, OVEN 53.3%, EDIS 76.7%, WebQA 90.0%). Across tasks it stays far above chance and above the majority baseline, which scores 0.0% because the held-out task's type is never the training fold's majority. LOTO accuracy is 71–80% at every layer from 0 to 27 and peaks at the final layer (82.3%); LOSO accuracy on supported tasks is 63–76% through layer 27 and 78.3% at the final layer. So, for this checkpoint and a linear probe, target type is no more decodable at an internal layer than at the final one, and the final-layer state carries type information shared across different strings. Much of it is already present in the final token: the embedding output, before any context, gives 73.6% (LOTO) and 68.1% (LOSO, supported), because tasks of one type tend to end their query the same way. Under LOTO, 23

Table 36: Reassignment arm per task, mean of two training seeds, 300 queries. Group: kept (trained on orig), reassigned (trained on rstr), unseen (no training source). neutral: mean accuracy under the four neutral paraphrases. Swap gap: orig minus swap.

Task	Type	Group	orig	rstr	orig − rstr	neutral	swap gap
HatefulMemes	CLS	kept	56.0	56.3	−0.3	53.8	1.0
ImageNet-1K	CLS	kept	64.5	52.5	+12.0	57.6	26.0
ImageNet-A	CLS	kept	39.3	31.2	+8.2	38.0	39.3
ImageNet-R	CLS	kept	64.2	42.2	+22.0	54.6	24.3
N24News	CLS	kept	66.3	61.8	+4.5	62.8	27.0
MSCOCO_i2t	RET	kept	50.8	46.3	+4.5	47.6	17.8
VisualNews_i2t	RET	kept	60.0	55.3	+4.7	59.4	17.8
VisualNews_t2i	RET	kept	62.3	59.0	+3.3	62.0	−1.2
WebQA	RET	kept	83.3	82.5	+0.8	82.5	4.8
ObjectNet	CLS	reassigned	57.8	54.5	+3.3	59.9	7.5
Place365	CLS	reassigned	32.2	33.0	−0.8	32.3	7.3
SUN397	CLS	reassigned	61.3	64.3	−3.0	58.9	1.8
VOC2007	CLS	reassigned	65.5	73.7	−8.2	56.7	8.7
A-OKVQA	VQA	reassigned	38.7	41.7	−3.0	40.8	1.2
ChartQA	VQA	reassigned	33.8	35.5	−1.7	35.5	−2.0
DocVQA	VQA	reassigned	68.5	71.2	−2.7	69.1	2.2
GQA	VQA	reassigned	47.3	44.8	+2.5	49.2	−4.3
InfographicsVQA	VQA	reassigned	31.5	32.5	−1.0	31.3	−0.2
OK-VQA	VQA	reassigned	46.5	46.7	−0.2	45.9	2.0
ScienceQA	VQA	reassigned	34.0	33.7	+0.3	34.2	−1.5
TextVQA	VQA	reassigned	60.3	59.0	+1.3	60.2	0.0
Visual7W	VQA	reassigned	39.3	41.3	−2.0	41.2	0.8
VizWiz	VQA	reassigned	24.5	26.3	−1.8	24.6	−0.5
CIRR	RET	reassigned	31.7	33.8	−2.2	32.2	6.3
MSCOCO_t2i	RET	reassigned	64.5	65.3	−0.8	65.3	−1.0
NIGHTS	RET	reassigned	65.7	66.5	−0.8	63.6	0.2
VisDial	RET	reassigned	54.3	56.8	−2.5	53.8	1.3
Country211	CLS	unseen	24.8	25.8	−1.0	26.1	14.8
EDIS	RET	unseen	62.7	62.7	+0.0	63.1	−0.8
FashionIQ	RET	unseen	11.7	11.0	+0.7	11.7	2.2
OVEN	RET	unseen	67.7	70.0	−2.3	68.5	1.5
Wiki-SS-NQ	RET	unseen	51.7	48.7	+3.0	48.9	3.2
MSCOCO	GND	unseen	49.2	46.7	+2.5	49.7	7.3
RefCOCO	GND	unseen	67.5	65.7	+1.8	68.0	3.2
RefCOCO-Matching	GND	unseen	70.5	70.3	+0.2	70.1	2.8
Visual7W-Pointing	GND	unseen	59.8	59.0	+0.8	59.1	0.2

tasks are classified perfectly and five more above 85%. The failures are the three WIKI tasks (EDIS and OVEN 0.0%, WebQA 23.3%), read mostly as IMAGE or ANSWER; Wiki-SS-NQ, which is unsupported; N24News (11.7%), read as ANSWER; the CROP tasks Visual7W-Pointing (28.3%) and MSCOCO (53.3%), read as LABEL or ANSWER; and FashionIQ (76.7%). Table 38 gives the LOSO confusion matrix. The only row that changes materially from LOTO is ANSWER, which LOSO leaves unsupported: with the shared VQA string held out, no ANSWER task remains in the training fold, and 538 of 600 VQA queries are read as LABEL.

Interpretation. Every string maps to exactly one target type, so no split can separate a template identifier from a type representation: within a task the probe can read either, and across strings it can read type cues that strings of one type share, beginning with their final token. The probe is therefore context for Section 4, not evidence for or against the task-ID account.

Instruction-free baseline. To measure how much of the type the query *content* carries on its own, we re-encoded the same 36 × 60 queries with the instruction deleted and probed the final layer (last column of Table 37; `code/layer_probe.py --qcond none`). Type stays far above chance:

Table 37: Target-type probe accuracy (%), 7 classes, 36 tasks × 60 queries. *Layer 0* is the embedding output, where the last-token state is that token's input embedding and carries no context. *No instr.* re-encodes the same queries with the instruction deleted (the *none* condition) and probes the final layer.

Split	Final layer (28)	Layer 0	No instr., final
Within-task, 30/30 split	96.3	—	82.9
LOTO, all 36 tasks	82.3	73.6	69.4
LOTO, 35 supported tasks	84.6	75.7	71.4
LOSO, all 36 tasks	54.4	47.3	41.2
LOSO, 25 supported tasks	78.3	68.1	59.3
Chance (uniform over 7)	14.3	14.3	14.3
Majority class of the training fold (LOTO)	0.0	0.0	0.0

Table 38: LOSO confusion matrix at the final layer, 36 tasks × 60 queries (rows: true type; columns: predicted). Correct counts on the diagonal.

True \ pred.	LABEL	ANSWER	CAPTION	IMAGE	DOCIMG	WIKI	CROP
LABEL (600)	**536**	61	0	3	0	0	0
ANSWER (600)	538	**0**	11	0	0	1	50
CAPTION (120)	1	3	**111**	5	0	0	0
IMAGE (360)	0	0	0	**345**	0	1	14
DOCIMG (60)	0	0	0	30	**0**	30	0
WIKI (180)	7	51	0	108	0	**14**	0
CROP (240)	26	43	0	3	0	0	**168**

LOTO 69.4% (with the instruction, 82.3%), LOSO on supported tasks 59.3% (78.3%), within-task 82.9% (96.3%). The same-layer contrast, 82.3% against 69.4% at the readout, is the measure of what the instruction adds; without it the per-layer LOTO curve is flat, 69–74% at every layer, whereas with it the curve rises from 73.6% at layer 0 to 82.3% (the two comparisons differ in depth as well as input, so the rise is not itself attributable to the instruction). Where the instruction adds is specific: LABEL (89.8% to 89.5%) and ANSWER (100% to 99.7%) tasks are unchanged, because a question or a class image carries its type in the content, whereas non-LABEL tasks whose query content does not fix the type collapse. NIGHTS falls from 100% to 0.0%, VisualNews_i2t and MSCOCO_i2t from 93.3% and 91.7% to 0.0%, VisualNews_t2i from 100% to 25.0%, CIRR from 100% to 43.3% and VisDial from 98.3% to 63.3%: every CAPTION query (120 of 120, all image-only) and 91 of 360 IMAGE queries are read as LABEL: an image with no text looks, to the probe, like a classification query. MSCOCO grounding improves (53.3% to 98.3%) once its eval-only wording is gone. The instruction's contribution to the readout is therefore concentrated in tasks whose content does not fix the type: image-only queries (NIGHTS and the two i2t tasks) and caption, dialogue or edit-text queries that could serve several targets (VisualNews_t2i, VisDial, CIRR). This overlaps with, but is not the same as, where the paraphrase audit locates its largest costs, the tasks whose only text is the instruction.

A second probe, used elsewhere. The type-enforcement pilot of Appendix B.18(ii) uses its own probe (`code/type_enforce.py`, same classifier, on the output embeddings of the shared-index run, `runs/type_enforce_all.json`). Its accuracies are also LOTO: 86.4% on queries and 60.1% on candidates, averaged over held-out tasks, with per-task query values from 100.0% (ImageNet-1K) down to 0.7% (N24News). None of these is a within-task accuracy; the only within-task numbers in this paper are the 96.3% and 82.9% above. Everything in this subsection is recomputed by `code/probe_reconcile.py` and `code/probe_string_split.py` (`runs/probe_reconcile.json`, `runs/probe_string_split.json`; with the suffix `_none` for the instruction-free capture).