LEND ME YOUR EYES: INSTRUCTION-AWARE TEXT EMBEDDINGS VIA ATTENTION RELAY

Anonymous authors
Paper under double-blind review

ABSTRACT

Text embedding models trained with contrastive learning learn to follow task instructions from instruction-paired data, while instruction-tuned LLMs already know how to follow them. We show that this instruction-following ability can carry over from an LLM to a Transformer-based embedder without any training. We propose Attention Relay, which passes the attention weights an LLM produces to the embedder's own attention. Across six instruction-tuned LLMs from the Qwen3, Llama 3.1 and OLMo 3 families and ten widely used embedding models that differ in tokenizer, size and pooling type, Attention Relay makes nearly every combination instruction-aware. Experiments that break the method down into its parts show that the LLM's attention weights track the instruction in its later layers and come largely from instruction tuning. They also show that relaying these weights selects which content in the text matters: it makes the aspect of the text that the instruction asks about dominant in the embedding, or restores that aspect where averaging had diluted it.

1 INTRODUCTION

Text embeddings are central to search, retrieval-augmented generation, clustering and classification (Muennighoff et al., 2023), and the success of Large Language Models (LLMs) has made them stronger: the best embedding models are now built on LLM backbones (Wang et al., 2024; BehnamGhader et al., 2024; Muennighoff et al., 2025; Zhang et al., 2025c). What makes two texts similar, however, can depend on the task: news articles can be grouped by topic or by where they happened, customer messages by how the customer feels or by what the customer needs. We call any model or readout that turns a text into one vector by pooling its token states an *embedder*; trained embedding models are the most common kind.

An embedder trained without instructions cannot follow one, even when the instruction is placed in its input. Most recent embedders are therefore trained with instructions: they take an instruction as part of their input and learn to use it by contrastive training on texts paired with instructions, from InstructOR (Su et al., 2023) to Qwen3-Embedding (Zhang et al., 2025c), which takes curated instruction data and a round of training for each embedder, to teach what the LLMs they are built on already do (Wei et al., 2022; Ouyang et al., 2022).

An instruction-tuned LLM follows such instructions readily, but it is no substitute for an embedder. Its hidden states were never shaped for similarity, and embeddings read from them without training, such as PromptEOL (Jiang et al., 2024), Echo embeddings (Springer et al., 2025) and Value Aggregation (Zhang et al., 2026), remain well behind trained embedders. Nor does the LLM's grasp of an instruction obviously carry over to an embedder, which differs from it in size, tokenizer and representation space.

We show that one part of an instruction-tuned LLM's instruction following does carry over, without any training: where the LLM reads a text when it is about to answer, the attention it pays to the text's tokens, which we call its *reading weights* (Figure 1). We propose *Attention Relay*, which hands these weights to the pooling of an instruction-agnostic embedder. The embedder never sees the instruction, and the embedding stays in the space the embedder was trained for.

We apply Attention Relay to instruction-tuned LLMs of the Qwen3, Llama 3.1 and OLMo 3 families (Yang et al., 2025a; Grattafiori et al., 2024; Team Olmo, 2025) and to widely used embedders, from

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

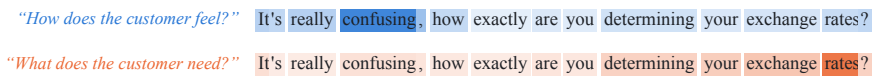


Figure 1: One customer message read by Qwen3-1.7B under two questions, each token shaded by its reading weight.

the sentence-transformers models all-MiniLM-L6-v2 and all-mpnet-base-v2 (Reimers & Gurevych, 2019) to BGE, GTE and E5 (Xiao et al., 2024b; Li et al., 2023; Wang et al., 2022) and Qwen3-Embedding (Zhang et al., 2025c), which differ in tokenizer, size and kind of pooling. We show that Attention Relay makes the embedders instruction-aware in nearly every combination of LLMs and embedders, on the New York Times (NYT) and IntentEmotion datasets and on InBedder's other sets (Peng et al., 2024).

2 Motivation

In this section, we show that an instruction-agnostic embedder holds what an instruction asks about but reads every text the same way, while an instruction-tuned LLM reads a text differently for each question but does not represent it for similarity.

2.1 An embedder holds what an instruction asks about but cannot weight it

An embedding is a weighted average over a text's tokens. Mean pooling gives every token the same weight, and CLS or last-token pooling lets one token attend to the others with attention the embedder learned for similarity in general. In an embedder trained without instructions, nothing in these weights responds to a question, and a question placed in the input is only more text. Asked whether a customer message is closer to one that shares its emotion or one that shares its intent, such an embedder follows the question in about half of the cases, as often as one that ignores it, while Qwen3-Embedding, trained with instructions, follows it in nearly all (Figure 2). A news article about a business deal in Japan belongs with business news when articles are grouped by topic and with news from Japan when they are grouped by country, yet such an embedder gives it one vector for both, although generic embeddings do encode what an instruction asks about (GSTransform; Feng et al., 2025). The embedder holds what an instruction asks about but has no way to weight it by the instruction.

2.2 An instruction-tuned LLM reads the text for the question

An instruction-tuned LLM given a text followed by a question computes every state of the text before it sees the question. Because of the causal mask, the text's states are the same under every question, and the question can act on the text only through where later positions attend. At the position where the LLM is about to answer, its attention over the text, summed over heads and renormalized over the text, is what we call its reading weights (Figure 1).

We measure how much the reading weights move with the question as the total variation distance between them under two questions about the same text, $\frac{1}{2}\sum_j |w_j(x, q_1) - w_j(x, q_2)|$, averaged over news articles asked for their topic or their location and customer messages asked for the customer's feeling or need. In every instruction-tuned LLM we examine, this *question sensitivity* stays near zero through the first half of the network, rises between 40 and 70% of its depth, and falls again at the last block (Figure 3). Instruction tuning largely creates this reading: the base checkpoints' weights move far less with the question at every block.

An instruction-tuned LLM does not, however, represent the text for similarity: its hidden states were never shaped for it, and embeddings read from them without training, such as PromptEOL (Jiang et al., 2024), Echo embeddings (Springer et al., 2025) and Value Aggregation (Zhang et al., 2026), remain well behind trained embedders. The LLM reads the text for the question but does not represent it for similarity; the embedder represents the text but cannot read it for the question.

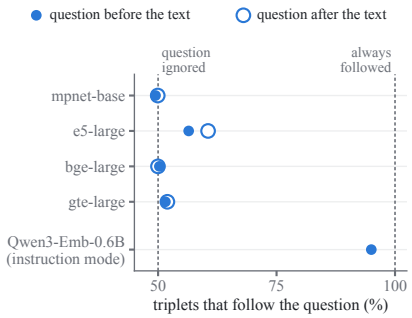


Figure 2: Question following in embedders without vs. with instruction training.

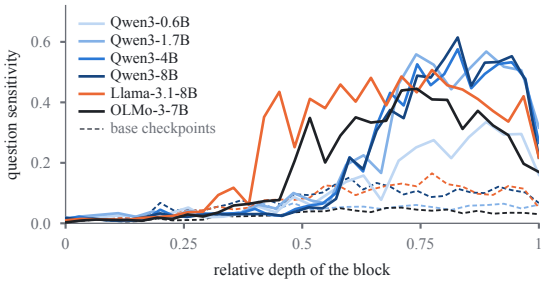


Figure 3: Question sensitivity at every block of each LLM (solid) and its base checkpoint (dashed).

3 Attention Relay

Attention Relay makes an instruction-agnostic embedder instruction-aware by giving it, for each text, the weights with which the LLM reads that text under the instruction (Figure 4).

3.1 Formulation

Reading weights. Let $x = (x_1, \ldots, x_n)$ be a text in the LLM's tokens and q an instruction. We give the LLM the text followed by the instruction in its chat template (Appendix A.2) and take the last position of the prompt, where the model is about to answer, which we call the *answer position*. As in PromptEOL (Jiang et al., 2024), the prompt asks for a one-word answer, so that the token the model is about to write is the answer itself rather than a lead-in such as "The". At block b, let $\alpha_j^h(x, q)$ be the attention from the answer position to text token x_j in head h. The LLM's *reading weights* are these attentions summed over the block's H heads and renormalized over the text:

$$w_j(x, q) \;=\; \frac{\sum_{h=1}^{H} \alpha_j^h(x, q)}{\sum_{k=1}^{n} \sum_{h=1}^{H} \alpha_k^h(x, q)}, \qquad j = 1, \ldots, n. \tag{1}$$

Because the text precedes the instruction, the causal mask keeps the text's states identical under every instruction; the instruction changes only where the answer position looks. The block b is set once per LLM and is the same for every embedder and dataset.

From the LLM's tokens to the embedder's. The embedder splits x into its own tokens, which need not match the LLM's. We carry w over through characters: each LLM token's weight is spread evenly over its non-space characters, and each embedder token i receives the weight of the characters it covers,

$$\tilde{w}_i \;\propto\; \sum_{j=1}^{n} w_j \, \frac{|C_i \cap C_j|}{|C_j|}, \tag{2}$$

where C_j and C_i are the non-space characters of LLM token j and embedder token i, and $\tilde{w}$ is renormalized over the embedder's text tokens. Tokens outside the text, such as special tokens or a fixed prefix, cover none of its characters and receive no weight.

Relaying the weights. Let r be the weights that the embedder's pooling places on its tokens. In mean pooling, r is uniform over the tokens and averages their final states. In CLS or last-token pooling, the embedding is the final state of one pooling token; at each layer, a head's attention row a from that token gives $r_i = a_i/m$ on the text tokens $i \in \mathcal{T}$, where $m = \sum_{i \in \mathcal{T}} a_i$ is the head's total attention on the text. Attention Relay multiplies r by the LLM's weights and renormalizes,

$$r'_i \;\propto\; r_i \, \tilde{w}_i^{\beta}, \tag{3}$$

with strength $\beta = 1$ unless stated otherwise; $\beta = 0$ returns the released embedder, and a larger β concentrates r' on the tokens the LLM weights most. The product keeps the embedder's own

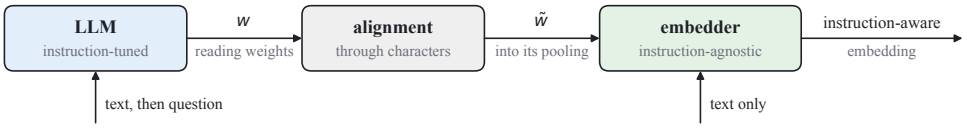


Figure 4: Attention Relay: the LLM's reading weights, carried through characters to the embedder's tokens, pool the embedder's own states.

preferences among the tokens the LLM reads and removes weight from the tokens it ignores. In mean pooling, r' becomes the pooling weights. In CLS or last-token pooling, every head at every layer spends its text attention m as $m\,r'$, keeps its attention outside the text,[1] and recomputes its output at the pooling token from the embedder's own values.

An equivalent logit form. In CLS or last-token pooling, the same change can be made before the softmax. Let s be the logits of a head's attention row at the pooling token, so that $a = \mathrm{softmax}(s)$. For any bias b over the positions,

$$\mathrm{softmax}(s+b)_i \;=\; \frac{e^{s_i}\,e^{b_i}}{\sum_k e^{s_k}\,e^{b_k}} \;=\; \frac{a_i\,e^{b_i}}{\sum_k a_k\,e^{b_k}}, \tag{4}$$

so adding b_i to a logit multiplies its weight by e^{b_i} and renormalizes. Choosing

$$b_i = \begin{cases} \beta \log \tilde{w}_i + c, & i \in \mathcal{T}, \\ 0, & i \notin \mathcal{T}, \end{cases} \qquad c = \log \frac{m}{\sum_{k\in\mathcal{T}} a_k\,\tilde{w}_k^{\beta}} \tag{5}$$

makes the denominator $e^c \sum_{k\in\mathcal{T}} a_k \tilde{w}_k^{\beta} + \sum_{k\notin\mathcal{T}} a_k = m + (1-m) = 1$, so each text token receives $e^c a_i \tilde{w}_i^{\beta} = m\,r'_i$ by equation 3 and every other position keeps a_i: the result is exactly the relayed row. The term $\beta \log \tilde{w}$ acts as a soft attention mask read from the LLM, and a token with $\tilde{w}_i = 0$ is masked outright. The constant c is the same for every text token in a head and holds the head's attention on the text at m; without it the bias would move attention from the text to the positions outside it, since $\tilde{w}_i \le 1$.

Alternative: the plain bias. We call equation 5 the *mass-preserving bias*. The *plain bias* sets $c = \beta \log |\mathcal{T}|$ in every head instead, so that uniform weights change nothing: it multiplies a head's attention on the text by $(|\mathcal{T}|\,\tilde{w}_i)^{\beta}$ and renormalizes the whole row, where the mass-preserving bias renormalizes within the text. Both give each head the same distribution r' over the text and differ only in how much of its attention the head keeps there: under the plain bias, the odds $m/(1-m)$ are multiplied by

$$\kappa \;=\; |\mathcal{T}|^{\beta} \sum_{i\in\mathcal{T}} r_i\,\tilde{w}_i^{\beta}, \tag{6}$$

which is 1 for uniform $\tilde{w}$, above 1 when the head already attends to the tokens the LLM weights, and below 1 when it attends to others. A head that agrees with the LLM then writes more from the text, and one that disagrees moves attention outside the text and writes less from it. The mass-preserving bias leaves how much each head writes to the embedder, so that only a distribution over the text passes from the LLM.

3.2 Why attention

Only the weights can cross. Of what an instruction-tuned LLM computes while it reads a text under an instruction, an embedder could be handed either vectors or weights. Vectors, whether hidden states, values or the representation of an answer, live in the LLM's own coordinates, with their own dimension and geometry, and cannot enter another model without a learned map. Weights over the text have no coordinates: they are a distribution over the text itself.

[1] In a causal embedder the first token serves as an attention sink (Xiao et al., 2024a); we count it outside the text, so its attention is kept as well.

The weights carry the instruction. They are also where the instruction acts. In a Transformer, attention is the only operation that moves information between positions, and with the text before the instruction, the text's values are the same under every instruction. The text's share of a head's output at the answer position, $\sum_{j=1}^{n} \alpha_j(x, q)\, v_j(x)$, therefore depends on the instruction only through its weights, so passing the weights passes everything the instruction changes in how the LLM reads the text, and nothing else.

The embedder already reads through weights. An embedder pools its text with a distribution over its tokens: uniform in mean pooling, its pooling token's attention in CLS or last-token pooling. Once the two tokenizations are aligned through characters, the LLM's weights mean the same thing in the embedder, how much each part of the text matters for the question, and they enter the embedder where its own weights already are. Which parts of a text matter for a question is a property of the text and the question, not of the model that reads them; how to represent those parts stays with the embedder.

4 Experiments

We evaluate Attention Relay across LLMs, embedders and kinds of instruction, then change one ingredient at a time and ask what the relay passes, what it can select and what it changes.

4.1 Setup

Models. We take the reading weights of six instruction-tuned LLMs: Qwen3 at 0.6B, 1.7B, 4B and 8B (Yang et al., 2025a), Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and OLMo-3-7B-Instruct (Team Olmo, 2025). Four of their base checkpoints serve as controls. We relay the weights into ten embedders that take no instruction: Qwen3-Embedding in its document mode (Zhang et al., 2025c), BGE (Xiao et al., 2024b), GTE (Li et al., 2023), E5 (Wang et al., 2022) and two sentence-transformers models (Reimers & Gurevych, 2019) (Appendix A.1). They span tokenizers, sizes and three kinds of pooling: the last token, [CLS] and the mean. We read each LLM at one block, chosen without labels where its weights move most with the question, and keep one setting per pair on every task. Unless stated otherwise, we relay Qwen3-1.7B's weights.

Tasks. We use InBedder's evaluation sets (Peng et al., 2024). NYT holds 2,144 news articles, labeled by topic and by the country where the news happened. We embed every article once under the topic question and once under the location question, cluster each set of embeddings by the aspect its question asks about, and report the harmonic mean of the two V-measures. IntentEmotion (IE) holds customer messages, labeled by how the customer feels and by what the customer needs, and arranges them in triplets. Each triplet asks whether a message is closer to one that shares its emotion or one that shares its intent; an embedding that ignores the question scores 100, and one that always follows it scores 200. Four more sets carry instructions of other kinds. FewRel, FewNerd and FewEvent ask every text for the type of a marked relation, entity or event, and InstructSTSB gives every pair of sentences its own instruction. We run them with Qwen3-1.7B here and with Llama and OLMo in Appendix C.1. Every gain comes with a 95% paired bootstrap interval (Appendix B).

4.2 Main results

Table 1 and Figure 5 show the main result: embedders that never see the question now embed a text according to it. Every embedder in Table 1 gains on NYT and IE, and nearly every pair of an LLM and an embedder gains in Figure 5. The relay also raises the scores on FewRel, FewNerd, FewEvent and InstructSTSB, where the instruction does not pick one of two aspects. Only e5-large-v2's gains on FewRel and FewEvent have intervals that include zero (Appendix B, Table 9). Qwen3-Embedding, trained with instructions and given the task, still scores higher on IE and the type sets but falls slightly below zero on InstructSTSB. We speculate that instruction training, in which one instruction serves a whole task, teaches an embedder to shift every embedding of the task rather than to pick out the words a question asks about.

Table 1: Five embedders that take no instruction, alone and relayed from Qwen3-1.7B (+ relay), and Qwen3-Embedding-0.6B trained with instructions (bottom row). IE is 100 when the question is ignored and 200 when it is always followed.

	Clustering, V-measure ↑				STS, Spearman ↑	Triplets ↑
Embedder	NYT	FewRel	FewNerd	FewEvent	InstructSTSB	IE
Qwen3-Embedding-0.6B (document mode)	55.9	54.4	42.7	59.1	0.0	100.0
+ relay	64.6	57.6	49.5	63.5	18.2	142.6
Gain	+8.7	+3.2	+6.8	+4.5	+18.2	+42.6
bge-large-en-v1.5	56.7	55.4	43.3	59.0	0.0	100.0
+ relay	67.2	58.8	48.0	64.3	20.3	157.8
Gain	+10.6	+3.4	+4.7	+5.3	+20.3	+57.8
gte-large-en-v1.5	56.8	54.1	42.3	56.8	0.0	100.0
+ relay	68.8	58.4	45.8	61.1	17.9	142.3
Gain	+12.0	+4.3	+3.5	+4.3	+17.9	+42.3
all-mpnet-base-v2	56.7	52.1	40.9	55.5	0.0	100.0
+ relay	66.3	55.8	45.9	61.7	11.6	135.6
Gain	+9.6	+3.7	+5.0	+6.2	+11.6	+35.6
e5-large-v2	54.3	56.4	41.4	61.1	0.0	100.0
+ relay	67.1	57.7	43.2	64.0	7.4	125.3
Gain	+12.8	+1.2	+1.8	+2.8	+7.4	+25.3
Qwen3-Embedding-0.6B (instruction mode)	68.4	63.4	58.1	67.3	−1.7	190.0

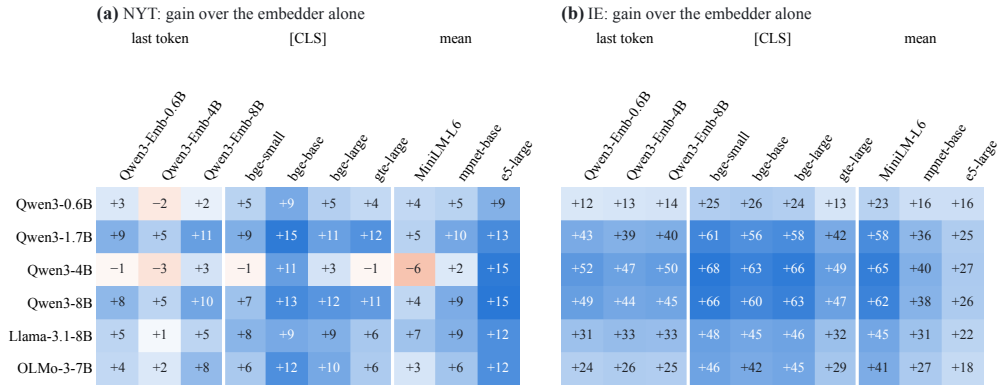


Figure 5: The gain of relaying for every pair of an instruction-tuned LLM and an embedder.

4.3 Ablations

On instruction tuning. Base checkpoints relay much less into the five embedders of Table 1 than their instruction-tuned siblings do (Figure 6). Read at their siblings' blocks, the weights of the four base checkpoints barely move with the question. The one case where a base's weights do move, Llama's on NYT, is also the one where a base relays nearly as much as its sibling.

On the LLM block. The gain tracks the question sensitivity through the LLM's depth (Figures 7 and 3). We relay the weights of every block of each LLM into the two embedders that pool by the mean, all-mpnet-base-v2 and e5-large-v2. Blocks where the question has not yet moved the weights give nothing, and the gain appears where the weights start to move, earliest in Llama. We can therefore choose the block without labels, as the one where the weights move most with the question; this choice loses little against the best block for every LLM but OLMo.

On the embedder's side. Relaying into the embedder's later layers gives nearly all of the gain, and into its earlier layers almost none (Appendix C.2). Replacing the mass-preserving bias with the plain bias moves the scores by about a point (Appendix C.3).

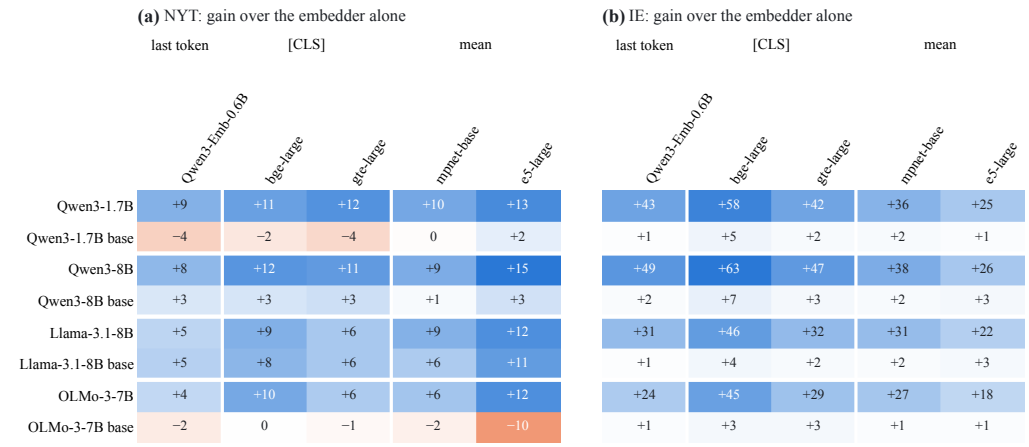


Figure 6: The gain of relaying from each instruction-tuned LLM and from its base checkpoint.

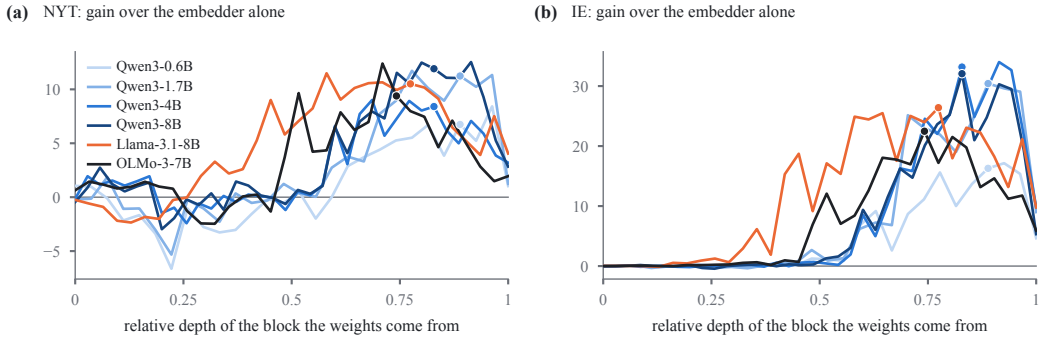


Figure 7: The gain of relaying from each block of each LLM; dots mark the blocks we read.

4.4 Interpretability of Attention Relay

What the relay passes. The gain comes from the words of each text that the LLM reads (Figure 8). Weights with the LLM's shape but the wrong words keep at most about a third of it, whether they are shuffled within the text, taken from another text or reduced to a profile of positions. Removing the tenth of words the LLM weights most takes away most of the gain, while removing a random tenth takes at most a quarter.

What the relay can select. The relay gains more when the text names what the question asks about. We split NYT's articles into those that name one of the ten countries or its people and those that do not, and cluster each set on its own. Where an article names the country, the location gain is two to six times larger than where the place is only implied, for four of the five embedders (Figure 9). Qwen3-Embedding, the one embedder built on an LLM, gains about as much in both.

What the relay changes. Relaying makes the asked aspect count in one of two ways: it surfaces an aspect the embedding held but did not cluster by, or makes an aspect more readable in the first place. A linear probe reads whether an embedding holds an aspect at all, and the V-measure whether the embedding clusters by it. An NYT article runs to hundreds of words, and most articles name the country or its people. Every embedding alone already holds the location, which a probe reads with 94 to 96% accuracy; relaying leaves the probe within a point but raises location V-measure by 15 to 27 points. A FewNerd text is one sentence of about 30 words, and its label is the type of one marked entity, often among entities of other types. Pooled over the whole sentence, that entity takes a small share of the embedding; relaying moves the pooling onto it and raises the probe's accuracy by 3 to 11 points. Figure 10 shows the difference under these two scenarios.

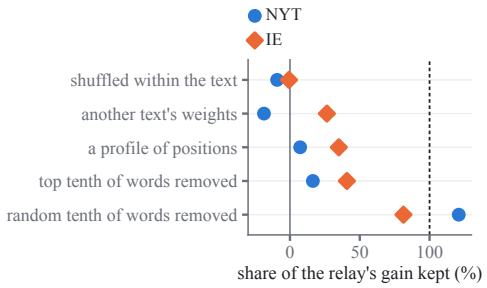


Figure 8: The share of the gain kept when the weights or the text change (bge-large).

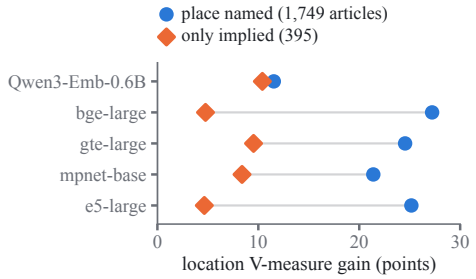


Figure 9: The location gain where an article names the country and where it does not.

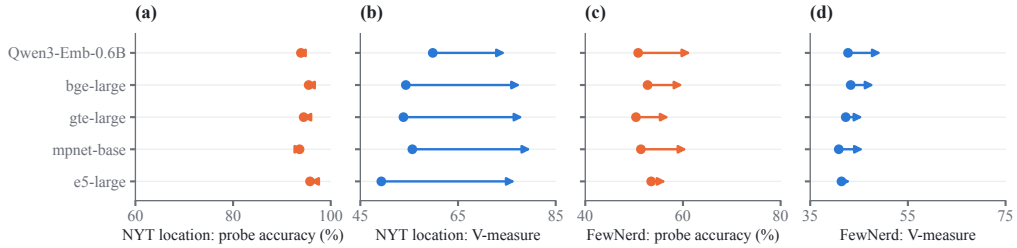


Figure 10: Probe accuracy and V-measure of the asked aspect, from each embedder alone to relayed.

5 Related Work

Instruction-aware text embeddings. A text can be grouped or compared in several valid ways, as studied in alternative clustering (Gondek & Hofmann, 2004; Niu et al., 2014) and conditional similarity (Veit et al., 2017), as in C-STS (Deshpande et al., 2023). Instruction-aware embedders let an instruction choose the way and learn to follow it in training: InstructOR (Su et al., 2023), E5-Mistral (Wang et al., 2024), GritLM (Muennighoff et al., 2025) and Qwen3-Embedding (Zhang et al., 2025c) through contrastive training with instructions, InBedder by learning to answer a question about the text (Peng et al., 2024), and GSTransform through a transformation of a generic embedder's outputs, trained for each instruction (Feng et al., 2025). Even so, retrieval models trained with instructions use them mostly as keywords (FollowIR; Weller et al., 2025). Attention Relay trains nothing: the instruction following comes from an LLM at inference, and the embedder is used as released.

Training-free embeddings from LLMs. An LLM's hidden state at the end of a prompt that asks for a text in one word serves as the text's embedding (PromptEOL; Jiang et al., 2024); later methods average several prompts (MetaEOL; Lei et al., 2024) or generated rewrites (GenEOL; Thirukovalluru & Dhingra, 2025), repeat the text (Echo embeddings; Springer et al., 2025), or pool the value vectors (Value Aggregation; Zhang et al., 2026). With a condition in the prompt, such readouts become conditional (PonTE; Yamada & Zhang, 2025), and self-contrastive steering improves them (Focusing Condition; Cheng et al., 2026). All of them represent the text in the LLM's own space; Attention Relay takes only the LLM's attention and keeps the space the embedder was trained for.

Attention as a question-conditioned signal. At the last position, attention heads fetch what a prediction needs from earlier tokens (Geva et al., 2023; Wang et al., 2023; Wu et al., 2025). As a score, this attention re-ranks passages in ICR (Chen et al., 2025) and QRHead (Zhang et al., 2025b), ranks documents once calibrated for position (Found in the Middle; Hsieh et al., 2024), decides which cached tokens to keep (SnapKV; Li et al., 2024b), and, taken from a small model, which tokens a larger model reads (SpecPrefill; Liu et al., 2025). Attention weights are not explanations in general (Jain & Wallace, 2019; Wiegreffe & Pinter, 2019), so we test the reading weights against shuffled, other-text and uniform weights. These uses stay within one model or one tokenizer family; Attention Relay carries the weights through characters into a different model.

Changing attention at inference. Attention steering scales a model's attention on one span, marked by the user (PASTA; Zhang et al., 2024a), found by the model (AutoPASTA; Zhang et al., 2024b) or holding the instruction, as in SpotLight (Venkateswaran & Contractor, 2026) and InstA-Boost (Guardieiro et al., 2025), by one factor. Image editing, as in Prompt-to-Prompt (Hertz et al., 2023) and Plug-and-Play (Tumanyan et al., 2023), injects the attention maps of one generation into another, which keeps its own values, and attention transfer (Zagoruyko & Komodakis, 2017; Li et al., 2024a) and MiniLM (Wang et al., 2020) train a student with a teacher's attention maps, also across tokenizers through character offsets (MATT; Haltiuk & Smywiński-Pohl, 2025).

6 Discussion

It selects; it cannot add. Relaying changes only the text's share of a head's output. At the pooling token, a head's output becomes

$$o' \;=\; m \sum_{i \in \mathcal{T}} r'_i \, v_i \;+\; \sum_{j \notin \mathcal{T}} a_j \, v_j, \tag{7}$$

with the embedder's own values and the same weight m on the text; in mean pooling, the embedding is $\sum_i r'_i u_i$ over the embedder's own final states u_i. Each relayed output is thus a convex combination of vectors the embedder computed itself: the instruction decides which of the embedder's content dominates, and only a distribution over the text, never a vector, passes from the LLM. What the embedder does not compute cannot be relayed: knowledge the LLM draws on to answer, such as the country a city lies in, and anything an instruction asks beyond which tokens matter, such as how two entities relate.

A shared backbone. Many embedders built on an LLM keep its weights and add a small fine-tuning change, often a LoRA adapter (Hu et al., 2022). If the LLM's instruction tuning were an adapter too, one backbone could serve as both models: with one adapter it would read the text for the question, and with the other it would embed it. The two would share their tokens, their heads and a single copy of the weights. Modern post-training, however, changes every weight of the model rather than training an adapter.

Open questions. The relay needs attention whose weights form a distribution over the text, as softmax attention does. Sparse attention, such as NSA (Yuan et al., 2025), reads only part of the text at full resolution, and linear attention, such as Gated DeltaNet (Yang et al., 2025b), keeps a state instead of weights; whether their implicit attention (Ali et al., 2025) can be relayed, from such an LLM or into such an embedder, is open. Nothing in the method is specific to text either: a vision-language model's attention over an image follows the question (Zhang et al., 2025a), and aligning image patches by area instead of characters would relay it into an image encoder's pooling; whether this makes image embeddings instruction-aware, and embeddings of other modalities after them, is the next test.

AI use statement

In this work, we used generative AI tools to write the code for the experiments and the analyses, to produce the figures and tables from the results files, to search and check the literature, and to draft sections of the paper. The research questions and the central idea of the method came from the authors. We have reviewed all AI-assisted work, and we take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Ameen Ali, Itamar Zimerman, and Lior Wolf. The hidden attention of Mamba models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1516–1534, 2025. doi: 10.18653/v1/2025.acl-long.76.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. Llm2vec: Large language models are secretly powerful text encoders. In *COLM*, 2024.

Shijie Chen, Bernal Jiménez Gutiérrez, and Yu Su. Attention in large language models yields efficient zero-shot re-rankers. In *International Conference on Learning Representations*, 2025. arXiv:2410.02642.

Zifeng Cheng, Lingyun Qian, Zhiwei Jiang, Cong Wang, Yafeng Yin, Fei Shen, Ao Zhou, and Qing Gu. Focusing condition: Inference-time self-contrastive steering elicits better conditional text embeddings in LLMs. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1135–1147, 2026. doi: 10.18653/v1/2026.acl-long.50.

Ameet Deshpande, Carlos Jimenez, Howard Chen, Vishvak Murahari, Victoria Graf, Tanmay Rajpurohit, Ashwin Kalyan, Danqi Chen, and Karthik Narasimhan. C-STS: Conditional semantic textual similarity. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5669–5690, 2023. doi: 10.18653/v1/2023.emnlp-main.345.

Yingchaojie Feng, Yiqun Sun, Yandong Sun, Minfeng Zhu, Qiang Huang, Anthony K. H. Tung, and Wei Chen. Don't reinvent the wheel: Efficient instruction-following text embedding based on guided space transformation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. arXiv:2505.24754.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12216–12235, 2023. doi: 10.18653/v1/2023.emnlp-main.751.

David Gondek and Thomas Hofmann. Non-redundant data clustering. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pp. 75–82, 2004. doi: 10.1109/ICDM.2004.10104.

Aaron Grattafiori, Abhimanyu Dubey, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Vitoria Guardieiro, Avishree Khare, Adam Stein, and Eric Wong. Instruction following by principled boosting attention of large language models. *arXiv preprint arXiv:2506.13734*, 2025.

Mykola Haltiuk and Aleksander Smywiński-Pohl. Model-aware tokenizer transfer. *arXiv preprint arXiv:2510.21954*, 2025.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross-attention control. In *International Conference on Learning Representations*, 2023.

Cheng-Yu Hsieh, Yung-Sung Chuang, Chun-Liang Li, Zifeng Wang, Long Le, Abhishek Kumar, James Glass, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. Found in the middle: Calibrating positional attention bias improves long context utilization. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 14982–14995, 2024. doi: 10.18653/v1/2024.findings-acl.890.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Sarthak Jain and Byron C. Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3543–3556, 2019. doi: 10.18653/v1/N19-1357.

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. Scaling sentence embeddings with large language models. In *Findings of EMNLP*, 2024.

Yibin Lei, Di Wu, Tianyi Zhou, Tao Shen, Yu Cao, Chongyang Tao, and Andrew Yates. Meta-task prompting elicits embeddings from large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10141–10157, 2024. doi: 10.18653/v1/2024.acl-long.546.

Alexander C. Li, Yuandong Tian, Beidi Chen, Deepak Pathak, and Xinlei Chen. On the surprising effectiveness of attention transfer for vision transformers. In *Advances in Neural Information Processing Systems*, volume 37, pp. 113963–113990, 2024a. doi: 10.52202/079017-3619.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970, 2024b.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.

Jingyu Liu, Beidi Chen, and Ce Zhang. Speculative prefill: Turbocharging TTFT with lightweight and training-free token importance estimation. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. In *EACL*, 2023.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. Generative representational instruction tuning. In *International Conference on Learning Representations*, 2025. arXiv:2402.09906.

Donglin Niu, Jennifer G. Dy, and Michael I. Jordan. Iterative discovery of multiple alternative clustering views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1340–1353, 2014. doi: 10.1109/TPAMI.2013.180.

Long Ouyang, Jeff Wu, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022. arXiv:2203.02155.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

Letian Peng, Yuwei Zhang, Zilong Wang, Jayanth Srinivasa, Gaowen Liu, Zihan Wang, and Jingbo Shang. Answer is all you need: Instruction-following text embedding via answering the question. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 459–477, 2024. doi: 10.18653/v1/2024.acl-long.27.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019. doi: 10.18653/v1/D19-1410.

Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 410–420, 2007.

Jacob Mitchell Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. Repetition improves language model embeddings. In *International Conference on Learning Representations*, 2025. arXiv:2402.15449.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of ACL*, 2023.

Team Olmo. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Raghuveer Thirukovalluru and Bhuwan Dhingra. GenEOL: Harnessing the generative power of LLMs for training-free sentence embeddings. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 2295–2308, 2025. doi: 10.18653/v1/2025.findings-naacl.122.

Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1921–1930, 2023.

Andreas Veit, Serge Belongie, and Theofanis Karaletsos. Conditional similarity networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1781–1789, 2017. doi: 10.1109/CVPR.2017.193.

Praveen Venkateswaran and Danish Contractor. Spotlight your instructions: Instruction-following with dynamic attention steering. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3752–3770, 2026. doi: 10.18653/v1/2026.eacl-long.174.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations*, 2023. arXiv:2211.00593.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. In *ACL*, 2024.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5776–5788, 2020.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022. arXiv:2109.01652.

Orion Weller, Benjamin Chang, Sean MacAvaney, Kyle Lo, Arman Cohan, Benjamin Van Durme, Dawn Lawrie, and Luca Soldaini. FollowIR: Evaluating and teaching information retrieval models to follow instructions. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 11926–11942, 2025. doi: 10.18653/v1/2025.naacl-long.597.

Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 11–20, 2019. doi: 10.18653/v1/D19-1002.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality. In *International Conference on Learning Representations*, 2025. arXiv:2404.15574.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *ICLR*, 2024a.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-Pack: Packed resources for general Chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 641–649, 2024b. doi: 10.1145/3626772.3657878.

Kosuke Yamada and Peinan Zhang. Out-of-the-box conditional text embeddings from large language models. *arXiv preprint arXiv:2504.16411*, 2025.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025b. arXiv:2412.06464.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, 2025. doi: 10.18653/v1/2025.acl-long.1126.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations*, 2017.

Jiarui Zhang, Mahyar Khayatkhoei, Prateek Chhikara, and Filip Ilievski. MLLMs know where to look: Training-free perception of small visual details with multimodal LLMs. In *International Conference on Learning Representations*, 2025a. arXiv:2502.17422.

Qingru Zhang, Chandan Singh, Liyuan Liu, Xiaodong Liu, Bin Yu, Jianfeng Gao, and Tuo Zhao. Tell your model where to attend: Post-hoc attention steering for LLMs. In *International Conference on Learning Representations*, 2024a. arXiv:2311.02262.

Qingru Zhang, Xiaodong Yu, Chandan Singh, Xiaodong Liu, Liyuan Liu, Jianfeng Gao, Tuo Zhao, Dan Roth, and Hao Cheng. Model tells itself where to attend: Faithfulness meets automatic attention steering. *arXiv preprint arXiv:2409.10790*, 2024b.

Wuwei Zhang, Fangcong Yin, Howard Yen, Danqi Chen, and Xi Ye. Query-focused retrieval heads improve long-context reasoning and re-ranking. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 23791–23805, 2025b. doi: 10.18653/v1/2025.emnlp-main.1214.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025c.

Yeqin Zhang, Yunfei Wang, Jiaxuan Chen, Ke Qin, Yizheng Zhao, and Cam-Tu Nguyen. LLM-based embeddings: Attention values encode sentence semantics better than hidden states. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. arXiv:2602.01572.

A IMPLEMENTATION DETAILS

A.1 SETTINGS

Every number in the paper comes from one setting per pair of an LLM and an embedder, used unchanged on every task. Tables 2 and 3 point to every model and dataset, and Table 5 gives each pair's setting.

Table 2: Every model in the paper.

Model	Hugging Face
Instruction-tuned LLMs	
Qwen3-0.6B	`Qwen/Qwen3-0.6B`
Qwen3-1.7B	`Qwen/Qwen3-1.7B`
Qwen3-4B	`Qwen/Qwen3-4B`
Qwen3-8B	`Qwen/Qwen3-8B`
Llama-3.1-8B-Instruct	`meta-llama/Llama-3.1-8B-Instruct`
OLMo-3-7B-Instruct	`allenai/Olmo-3-7B-Instruct`
Base checkpoints	
Qwen3-1.7B-Base	`Qwen/Qwen3-1.7B-Base`
Qwen3-8B-Base	`Qwen/Qwen3-8B-Base`
Llama-3.1-8B	`meta-llama/Llama-3.1-8B`
OLMo-3-7B	`allenai/Olmo-3-1025-7B`
Embedders, used without an instruction	
Qwen3-Embedding-0.6B	`Qwen/Qwen3-Embedding-0.6B`
bge-large-en-v1.5	`BAAI/bge-large-en-v1.5`
gte-large-en-v1.5	`Alibaba-NLP/gte-large-en-v1.5`
all-mpnet-base-v2	`sentence-transformers/all-mpnet-base-v2`
e5-large-v2	`intfloat/e5-large-v2`
all-MiniLM-L6-v2	`sentence-transformers/all-MiniLM-L6-v2`
bge-small-en-v1.5	`BAAI/bge-small-en-v1.5`
bge-base-en-v1.5	`BAAI/bge-base-en-v1.5`
Qwen3-Embedding-4B	`Qwen/Qwen3-Embedding-4B`
Qwen3-Embedding-8B	`Qwen/Qwen3-Embedding-8B`

Table 3: Every dataset, from InBedder's evaluation (Peng et al., 2024).

Dataset	Hugging Face
Used throughout	
NYT	`BrandonZYW/NYTClustering`
IntentEmotion (IE)	`BrandonZYW/IntentEmotion`
Used in §4	
FewRel	`BrandonZYW/FewRelClustering`
FewNerd	`BrandonZYW/FewNerdClustering`
FewEvent	`BrandonZYW/FewEventClustering`
InstructSTSB	`BrandonZYW/InstructSTSB`

The LLMs. Each text is cut to its first 400 tokens in the LLM's tokenizer and given to the LLM in its chat template (Appendix A.2); a base checkpoint is given its instruct sibling's template. The reading weights are taken at the block of Table 4, counting from 0: the block with the highest question sensitivity (§2.2) in the second half of the network, excluding the last block, with the sensitivity averaged over the texts of NYT and IE and no labels read. A base checkpoint is read at its sibling's block.

The embedders. Each embedder reads the same text, cut as above, and no instruction. All relay the LLM's weights in the multiply form with $\beta = 1$ (equation 3), and those that pool with attention do so with the mass-preserving bias in every head of every layer. The weights reach the embedder's tokens through characters (equation 2); the first token of Qwen3-Embedding, the attention sink

Table 4: Each LLM, the block its reading weights are taken at.

LLM	Blocks	Block read
Qwen3-0.6B	28	24
Qwen3-1.7B	28	24
Qwen3-4B	36	29
Qwen3-8B	36	29
Llama-3.1-8B-Instruct	32	24
OLMo-3-7B-Instruct	32	23
Qwen3-1.7B-Base	28	24
Qwen3-8B-Base	36	29
Llama-3.1-8B	32	24
OLMo-3-7B	32	23

of its document mode, is counted outside the text. e5-large-v2 reads every text after the prefix `"query: "`, the input format its model card prescribes for clustering and classification; like the special tokens, the prefix lies outside the text and gets no relayed weight.

Table 5: The setting of each embedder, the same with every LLM.

Embedder	Pooling	Relayed into	Max. tokens
Qwen3-Embedding-0.6B	last token	attention	–
Qwen3-Embedding-4B	last token	attention	–
Qwen3-Embedding-8B	last token	attention	–
bge-large-en-v1.5	[CLS]	attention	512
bge-small-en-v1.5	[CLS]	attention	512
bge-base-en-v1.5	[CLS]	attention	512
gte-large-en-v1.5	[CLS]	attention	512
all-mpnet-base-v2	mean	pooling weights	512
e5-large-v2	mean	pooling weights	512
all-MiniLM-L6-v2	mean	pooling weights	512

A.2 CHAT TEMPLATE

Every LLM reads every text in its own chat template as released, with the template's default system message, if any, and thinking turned off where the template offers it. The text and the task's instruction (Table 6) fill the user turn, and the answer position is the final token of the sequence. For Qwen3, whose template is shown below, that token is the line break after `</think>`.

```
<|im_start|>user
Text:
{text}

{instruction}

Answer in one word.<|im_end|>
<|im_start|>assistant
<think>

</think>
```

The sequence is tokenized piece by piece (the template up to `Text:`, the text, the instruction, the closing sentence and the template's end), so that no token crosses the text's boundaries.

Table 6: The instruction each dataset puts in the template's instruction slot.

Dataset	Instruction
NYT, topic	What is the topic of news?
NYT, location	Where did the news happen?
IntentEmotion, emotion	How does the customer feel?
IntentEmotion, intent	What does the customer need?
FewRel	Here is a sentence. Please tell me the relation type between two specified entities appended after the sentence.
FewNerd	Here is a sentence. Please tell me the type of the specified entity appended after the sentence.
FewEvent	Here is a sentence. Please tell me the type of the specified event according to the trigger words appended after the sentence.
InstructSTSB	each pair's own instruction, from the dataset

A.3 EMBEDDER INPUTS

Every embedder reads the text as its own tokenizer splits it, with its own special tokens. Table 7 gives what each embedder reads around the text in each condition, where `{text}` is the text, `{question}` the question or the set's instruction of Table 6, and `{task}` a task description of Table 8.

Table 7: What each embedder reads around the text, verbatim; `\n` is a line break.

Embedder	Condition	Input
Every embedder but e5-large-v2	alone and relayed	`{text}`
e5-large-v2	alone and relayed	`query: {text}`
all-mpnet-base-v2, bge-large, gte-large	the question in the input, before	`{question} {text}`
	or after the text (Figure 2)	`{text} {question}`
e5-large-v2	the same	`query: {question} {text}`
		`query: {text} {question}`
bge-large, gte-large, all-mpnet-base-v2	the set's instruction in the input (Table 10)	`{question} {text}`
e5-large-v2	the same	`query: {question} {text}`
Qwen3-Embedding-0.6B, instruction mode	the other sets' instructions (Tables 1 and 10)	`Instruct: {question}\n Query:{text}`
Qwen3-Embedding-0.6B, instruction mode	NYT and IE (Table 1 and Figure 2)	`Instruct: {task}\n Query:{text}`

Table 8: The task descriptions Qwen3-Embedding-0.6B reads in its instruction mode for NYT and IE, verbatim.

Question	Task description
NYT, topic	Identify the topic of the given news article
NYT, location	Identify the location where the given news article happened
IE, emotion	Identify the emotion the customer expresses in the given message
IE, intent	Identify what the customer needs in the given message

A.4 CONTROLS

Figure 8 asks whether the gain needs the LLM's weights on the text's own words. Three controls keep the shape of the weights but cut their tie to the words. *Shuffled within the text* permutes the text's weights at random over its own tokens. *Another text's weights* gives the text, token by token, the weights of a random text of the same length, or the nearest length. *A profile of positions* gives each token the LLM's average weight at the same relative position, under the same question, over all the other texts, with relative position split into twenty bins. It keeps where in a text the LLM tends to

look, but not which words this text holds. The shuffled and other-text controls average three random draws. Two more controls remove words from each text before the embedder reads it: the tenth of its words that the LLM weights most under the question, or a random tenth. On the shortened texts, we relay the LLM's weights on the remaining words and compare with the embedder alone on the same texts.

B Evaluation metrics

Let $e_q(x)$ be an embedder's embedding of the text x under the question q; the embedder alone gives the same embedding under every question.

V-measure. For the class labels Y and a clustering K of the same texts (Rosenberg & Hirschberg, 2007),

$$h = 1 - \frac{H(Y \mid K)}{H(Y)}, \qquad c = 1 - \frac{H(K \mid Y)}{H(K)}, \qquad V = 100 \cdot \frac{2\,h\,c}{h + c}, \tag{8}$$

where H is the entropy of the empirical distribution. K is k-means with as many clusters as classes, on the centered, unit-length embeddings, and V is averaged over ten initializations.

NYT. With V_{topic} the V-measure of the topic labels on the embeddings under the topic question, and V_{loc} that of the country labels on the embeddings under the location question,

$$\text{NYT} = \frac{2\,V_{\text{topic}}\,V_{\text{loc}}}{V_{\text{topic}} + V_{\text{loc}}}. \tag{9}$$

IntentEmotion. For each question $q \in \{\text{emotion}, \text{intent}\}$, a triplet $(a, x^+, x^-) \in T_q$ holds an anchor message a, a message x^+ that shares only the anchor's answer to q, and a message x^- that shares only its answer to the other question:

$$\text{IE} = 100 \sum_q \frac{1}{|T_q|} \sum_{(a,x^+,x^-) \in T_q} \mathbf{1}\big[\cos(e_q(a), e_q(x^+)) > \cos(e_q(a), e_q(x^-))\big]. \tag{10}$$

The two sets hold the same messages with x^+ and x^- swapped, so an embedding that ignores the question scores 100, and one that always follows it scores 200. Figure 2 reports IE/2, the share of triplets that follow the question. We use the triplets of the first 2,000 anchors in sorted order, 2,038 per question.

InBedder's other sets. FewRel, FewNerd and FewEvent report the V-measure of their type labels. InstructSTSB reports $100\,\rho$, where ρ is the Spearman correlation, over the sentence pairs (s_1, s_2) with instruction q and binary label y, between $\cos(e_q(s_1), e_q(s_2))$ and y.

Linear probe. The probe accuracy is the five-fold stratified cross-validated accuracy of a logistic regression ($C = 1$; Pedregosa et al., 2011) on the centered, unit-length embeddings; its chance level is the share of the largest class.

Gains and intervals. A gain is $\Delta = S_{\text{relay}} - S_{\text{alone}}$, a score of the relayed embedder minus the same score of the embedder alone. The share of the gain that a control of Figure 8 keeps is $100\,\Delta_{\text{control}}/\Delta$, where Δ_{control} is the gain under the control (Appendix A.4). A bootstrap replicate b resamples the items (texts, triplets or sentence pairs) with replacement and scores both embeddings on the same resample, giving Δ_b^*; the 95% interval runs from the 2.5th to the 97.5th percentile of Δ_b^*. Table 9 gives every gain of Table 1 with its interval.

Table 9: The gains of Table 1, with their 95% paired bootstrap intervals.

Set	Qwen3-Emb-0.6B	bge-large	gte-large	mpnet-base	e5-large
NYT	+8.7 [2.9, 12.2]	+10.6 [7.5, 16.2]	+12.0 [6.7, 16.0]	+9.6 [6.0, 14.8]	+12.8 [8.8, 20.2]
FewRel	+3.2 [1.5, 5.0]	+3.4 [1.8, 5.0]	+4.3 [2.4, 5.8]	+3.7 [1.8, 4.5]	+1.2 [−0.6, 3.2]
FewNerd	+6.8 [4.4, 8.2]	+4.7 [2.1, 6.0]	+3.5 [1.3, 4.9]	+5.0 [2.8, 6.2]	+1.8 [0.1, 3.7]
FewEvent	+4.5 [2.4, 6.4]	+5.3 [3.1, 6.7]	+4.3 [2.3, 5.7]	+6.2 [3.4, 7.8]	+2.8 [−0.3, 5.6]
InstructSTSB	+18.2 [16.6, 20.0]	+20.3 [18.5, 22.1]	+17.9 [16.3, 19.5]	+11.6 [10.6, 12.7]	+7.4 [6.7, 8.1]
IE	+42.6 [40.2, 45.1]	+57.8 [54.9, 60.6]	+42.3 [39.7, 44.9]	+35.6 [33.2, 37.9]	+25.3 [23.2, 27.5]

C Additional experiments

C.1 InBedder's other sets with other LLMs

Table 10 repeats InBedder's other sets with the weights of Llama-3.1-8B-Instruct and OLMo-3-7B-Instruct beside Qwen3-1.7B's, and adds each embedder given the instruction in its input. OLMo's weights give the same picture as Qwen3-1.7B's; with Llama's, FewRel falls for every embedder. Given the instruction in its input, every embedder scores below zero on InstructSTSB.

C.2 On the embedder's layers

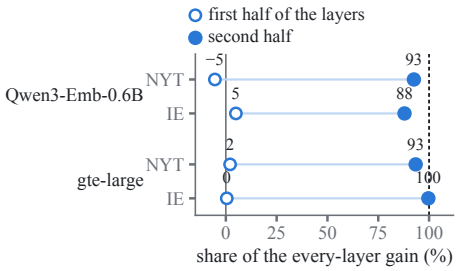


Figure 11: Relaying into half of the embedder's layers, as a share of the every-layer gain.

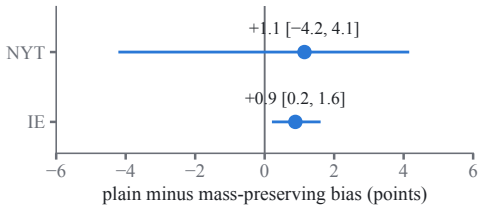


Figure 12: The plain bias minus the mass-preserving bias, with its 95% interval (bge-large).

The embedding keeps only what the embedder's later layers read. Qwen3-Embedding-0.6B and gte-large pool with attention at every layer, so the weights can be relayed into some of their layers and not others: relayed into the second half alone, they give nearly the whole gain, and into the first half alone almost none (Figure 11), although the pooling token attends to the text at least as much in the first half as in the second. Averaged over the texts, the pooling token of the embedder alone puts 0.32 of its attention on the text in the first half of Qwen3-Embedding-0.6B's layers and 0.30 in the second on NYT (IE: 0.24 and 0.23), and 0.81 and 0.71 in gte-large's (IE: 0.53 and 0.42).

C.3 On the text share

We compare the mass-preserving bias with the plain bias defined in §3.1. What the LLM contributes is where each head reads within the text, not how much of its attention goes to the text. The mass-preserving bias holds each head's attention on the text fixed, while the plain bias also lets it grow in the heads that agree with the LLM and shrink in the others; switching between them changes either score by about a point, a small fraction of the gain (Figure 12).

C.4 Cost

Measured on one H100 at the runs' batch sizes, over 256 NYT articles under the location question, a full pass of Qwen3-1.7B costs 4.6 ms per article and question, and relaying adds at most 0.3 ms to the embedder's own pass, which takes 0.7 to 5.9 ms (Table 11).

Table 10: InBedder's other sets with the weights of Qwen3-1.7B (a), Llama-3.1-8B-Instruct (b) and OLMo-3-7B-Instruct (c): each embedder alone, relayed, and given the instruction in its input.

(a) Qwen3-1.7B

Embedder	FewRel	FewNerd	FewEvent	InstructSTSB
Qwen3-Embedding-0.6B	54.4 → 57.6	42.7 → 49.5	59.1 → 63.5	0.0 → 18.2
bge-large-en-v1.5	55.4 → 58.8	43.3 → 48.0	59.0 → 64.3	0.0 → 20.3
gte-large-en-v1.5	54.1 → 58.4	42.3 → 45.8	56.8 → 61.1	0.0 → 17.9
all-mpnet-base-v2	52.1 → 55.8	40.9 → 45.9	55.5 → 61.7	0.0 → 11.6
e5-large-v2	56.4 → 57.7	41.4 → 43.2	61.1 → 64.0	0.0 → 7.4
Given the instruction in the input (Qwen3-Embedding: its instruction mode)				
Qwen3-Embedding-0.6B	63.4	58.1	67.3	−1.7
bge-large-en-v1.5	56.1	44.6	61.5	−13.7
gte-large-en-v1.5	53.0	43.7	60.4	−8.5
all-mpnet-base-v2	56.2	46.9	64.9	−11.2
e5-large-v2	60.3	45.8	68.0	−11.8

(b) Llama-3.1-8B-Instruct

Embedder	FewRel	FewNerd	FewEvent	InstructSTSB
Qwen3-Embedding-0.6B	54.4 → 51.7	42.7 → 46.7	59.1 → 61.7	0.0 → 21.9
bge-large-en-v1.5	55.4 → 52.0	43.3 → 44.4	59.0 → 62.7	0.0 → 23.0
gte-large-en-v1.5	54.1 → 47.9	42.0 → 41.2	56.8 → 59.5	0.0 → 20.7
all-mpnet-base-v2	52.1 → 50.9	40.9 → 41.8	55.8 → 58.5	0.0 → 14.1
e5-large-v2	56.4 → 55.7	41.5 → 41.7	61.1 → 62.8	0.0 → 8.9
Given the instruction in the input (Qwen3-Embedding: its instruction mode)				
Qwen3-Embedding-0.6B	63.4	57.8	67.3	−1.7
bge-large-en-v1.5	56.6	44.9	62.1	−13.6
gte-large-en-v1.5	53.0	43.6	60.0	−8.5
all-mpnet-base-v2	56.2	47.0	64.8	−11.2
e5-large-v2	60.5	45.8	68.3	−11.8

(c) OLMo-3-7B-Instruct

Embedder	FewRel	FewNerd	FewEvent	InstructSTSB
Qwen3-Embedding-0.6B	54.4 → 58.9	42.7 → 50.7	59.1 → 60.8	0.0 → 24.5
bge-large-en-v1.5	55.4 → 59.3	43.1 → 50.1	59.0 → 61.3	0.0 → 25.2
gte-large-en-v1.5	54.1 → 59.1	42.2 → 47.7	56.8 → 59.4	0.0 → 23.2
all-mpnet-base-v2	52.1 → 56.6	40.9 → 48.1	55.8 → 59.1	−0.0 → 15.8
e5-large-v2	56.4 → 57.9	41.3 → 44.2	61.1 → 63.0	0.0 → 9.9
Given the instruction in the input (Qwen3-Embedding: its instruction mode)				
Qwen3-Embedding-0.6B	63.4	58.0	67.3	−1.7
bge-large-en-v1.5	56.6	44.8	62.1	−13.6
gte-large-en-v1.5	53.0	43.4	60.2	−8.5
all-mpnet-base-v2	56.1	47.0	64.8	−11.2
e5-large-v2	60.3	45.9	68.3	−11.8

Table 11: Milliseconds per text and question on one H100: the full pass of Qwen3-1.7B, and each embedder alone and relayed.

Model	alone	relayed
Qwen3-1.7B, the full pass	4.58	
Qwen3-Embedding-0.6B	2.28	2.61
bge-large-en-v1.5	0.70	0.98
gte-large-en-v1.5	1.13	1.40
all-mpnet-base-v2	2.58	2.56
e5-large-v2	5.88	5.99