General-Purpose Instruction-Aware Text Embeddings via Dual-Level Contrastive Learning

Anonymous ACL submission

Abstract

Text embedding models are typically trained in an instruction-agnostic manner, limiting their ability to follow explicit user instructions in retrieval tasks. We propose a unified contrastive learning framework for instruction-aware text embeddings based on Dual-Level Instructional Contrastive Learning (DICL), with an Asymmetric Fusion Distillation strategy to preserve general semantic capability. We introduce GEDI, a large-scale instruction–text dataset, and ICE, a benchmark for evaluating instruction-following behavior in embedding models. Experiments demonstrate that our method achieves state-of-the-art performance on ICE and improves instruction-following retrieval by up to double-digit p-MRR gains across multiple benchmarks and embedding backbones, while retaining 99.6% of the base model's performance on standard embedding benchmarks.

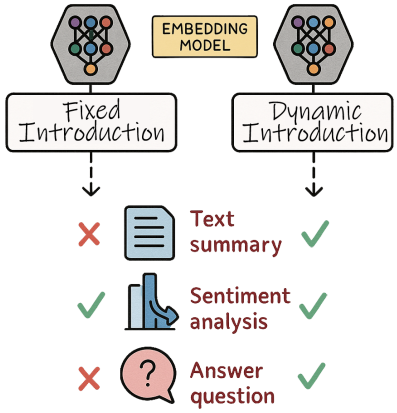

Figure 1: An illustration comparing fixed and dynamic instruction strategies for embedding models. The model on the left uses a fixed instruction and is limited to a specific task. The model on the right supports dynamic instructions and can handle various tasks (text summarization, sentiment analysis, question answering, etc.) based on the user's instruction.

1 Introduction

Imagine a user interacting with a digital assistant by providing the same input text under different natural-language instructions, such as *"Explain this article."* versus *"What is the sentiment of this paragraph?"*. Although the text remains unchanged, the intended tasks are fundamentally different. A well-designed embedding model should therefore produce instruction-dependent representations that reflect the user's intent. In practice, however, most existing text embeddings ignore explicit instructions and map each input text to a single fixed vector. As instruction-driven conversational and retrieval systems become increasingly common, this mismatch highlights a critical challenge: embedding models must follow user instructions while preserving their general-purpose semantic capability.

Text embeddings are foundational for modern NLP applications, supporting semantic search, retrieval-augmented generation (RAG) (Palangi et al., 2016), and knowledge-intensive tasks (Subercaze et al., 2015). However, existing methods struggle to align embeddings with diverse user instructions. As illustrated in Figure 1, many approaches rely on fixed, templated prompts that bind models to specific task contexts, limiting their ability to generalize across dynamic instructions such as summarization, sentiment analysis, and question answering. Even when instructions are explicitly provided, traditional models often fail to incorporate them meaningfully, treating different instructions as noise rather than informative context (Figure 5). Moreover, naïvely fine-tuning on instruction data risks catastrophic forgetting, degrading the model's original semantic capabilities (Zheng et al.). Collectively, these issues reveal three key challenges: (1) reliance on static templates instead of open-ended instruction handling, (2) inadequate modeling of instruction–content interactions, and (3) the absence of systematic evaluation for instruc-

tion understanding.

We propose a unified framework for learning *general-purpose instruction-aware text embeddings* via dual-level contrastive learning. Our framework introduces four key components. **Dual-Level Instructional Contrastive Learning (DICL)** explicitly models instruction-conditioned relevance while preserving instruction-invariant semantic structure, enabling embeddings to adapt to different instructions without fragmenting the representation space. **Asymmetric Fusion Distillation (AFD)** further stabilizes training by selectively transferring task-agnostic knowledge from the base encoder, ensuring strong general-purpose embedding capability. To support effective training and evaluation, we construct **GEDI**, a large-scale dataset of naturally phrased instruction–text pairs, and introduce **ICE**, the first benchmark designed for systematic evaluation of instruction-following behavior in embedding models. Extensive experiments demonstrate that our framework achieves state-of-the-art instruction-following performance on ICE and yields substantial improvements across multiple instruction-aware retrieval benchmarks, while retaining 99.6% of the base model's performance on MTEB.

2 Related Work

Instruction-Augmented Embedding Models. Recent models integrate natural language instructions to produce task-adaptive embeddings. Approaches include TART (Asai et al., 2022) and INSTRUCTOR (Su et al., 2023), which employ multi-task instruction tuning, as well as LLM-Embedder (Chen et al., 2024) and task-focused embeddings (Li et al., 2024) that categorize instructions for retrieval. While effective, these methods often rely on predefined templates or fixed task categories, limiting their adaptability to diverse or unseen instructions. Our work avoids rigid templates and explicitly models instruction semantics through contrastive learning.

Contrastive Learning Frameworks. Contrastive learning improves embeddings by distinguishing positive and negative pairs. Techniques such as improved negative sampling in RocketQA (Qu et al., 2021) and hard-negative methods in HNCSE (Liu et al., 2024) have advanced generic semantic similarity. However, these methods do not explicitly account for instruction context. In contrast, our DICL framework directly incorporates instruction–input pairs into the contrastive objective to learn instruction-sensitive representations.

LLM-Driven Embedding Optimization. LLMs have been used to generate synthetic data and supervise embedding training, as seen in multilingual dataset creation (Wang et al., 2024) and retrieval-generation combined approaches (Zhang et al., 2023). While promising, such methods often depend heavily on LLM inference and do not directly model instruction semantics within embeddings. Our approach leverages existing instruction-following data in a contrastive setup to naturally align instructions with content.

Model Fusion and Distillation. Techniques like model fusion and distillation aim to add capabilities without losing existing performance. Examples include RetroMAE with self-distillation (Chen et al., 2024) and mixture-of-experts designs (Guo et al., 2025). Inspired by these, our Asymmetric Fusion Distillation blends the original and instruction-tuned embeddings to preserve general semantics while enhancing instruction sensitivity.

3 Methods

3.1 GEDI: General Embedding Data with Instructions

To train our instruction-aware embeddings, we curate a large dataset of instruction–text–output triplets, capturing realistic user queries. We start from the Alpaca-CoT collection of instruction-following data (which contains many question–answer pairs with instructions). After extracting examples with non-empty `instruction`, `input`, and `output` fields (all in English), we filter for practical length constraints (4–1024 characters for each field), yielding about 3 million candidates. Figure 2 shows the character-length distribution of instructions, inputs, and outputs after initial filtering.

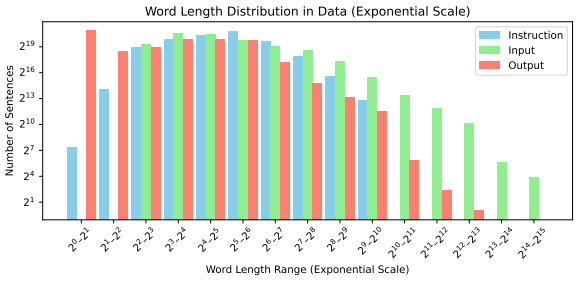

Figure 2: Word length distribution of Instructions, Inputs, and Outputs in the Alpaca-CoT dataset (3M entries after initial filtering).

Many of these samples are noisy or redundant. To ensure high quality, we design a multi-expert voting framework for further filtering (Figure 3). First, we remove duplicates within each field and across fields (e.g., identical input and instruction). Next, we use multiple pre-trained models to score the semantic similarity between each (instruction, input) pair and the output. We compute similarity scores for each model and use the median score as a threshold: samples above the threshold "pass" for that model. We then apply a logical AND across models, keeping only samples that pass every expert. This two-stage filtering yields about 54,449 high-quality examples.

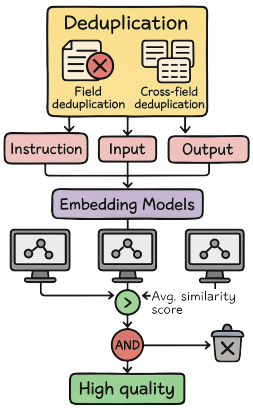

Figure 3: Multi-expert binary voting framework for automatic data filtering. The framework integrates deduplication and model-wise semantic similarity evaluation, retaining samples that meet quality thresholds across all experts.

To ensure quality, we employed Deepseek-R1 to verify that each example's output correctly follows its instruction, resulting in 22,026 well-aligned instruction-text pairs in our GEDI dataset. We further generated hard-negative examples using the same model to create semantically related but incorrect outputs for contrastive training. Comparative analysis against existing collections—MEDI (Su et al., 2023), InBedder (Peng et al., 2024), and InF-IR (Zhuang et al., 2025)—demonstrates GEDI's superior diversity and coverage. As shown in Table 1, GEDI yields more clusters (9) and a higher semantic diversity score (0.03) than the other datasets. While InF-IR achieves a high Silhouette score (0.67), indicating well-separated clusters, it covers fewer semantic categories (3 clusters). In contrast, GEDI maintains a balanced profile with broader topic coverage, as reflected in its higher cluster count and competitive CH/log(n) score, confirming its more comprehensive and varied instruction-task distribution.

Metric	MEDI	Inbedder	InF-IR	GEDI
DBSCAN	3	6	3	9
Noise Points	0.00	98.00	13.00	220.00
Silhouette	-0.26	0.04	0.67	0.31
Semantic	0.01	0.01	0.01	0.03
Adaptive EPS	0.81	1.06	0.75	0.95
CH / log(n)	0.21	22.49	1160.80	401.62
Noise Rate %	0.00	0.98	0.13	2.20
Sample Size	319	10000	10000	10000

Table 1: Cluster Analysis Results Showing Partition Quality, Diversity Measures and Algorithm Performance Across Datasets. CH Index is normalized as CH/log(n) to mitigate sample size effects.

Figure 4 provides a qualitative comparison of instruction semantics. We encode each instruction from the datasets by prompting `"What is the functional category of the given instruction?"` using `roberta-large-InBedder` (Peng et al., 2024), then project the vectors via PCA. MEDI instructions cluster tightly in one region, suggesting limited variety. InBedder has more spread but still concentrates to one side. In contrast, GEDI covers a broad, balanced space (pink points), reflecting a wider range of instruction types. This diversity makes GEDI a strong foundation for training instruction-aware embeddings.

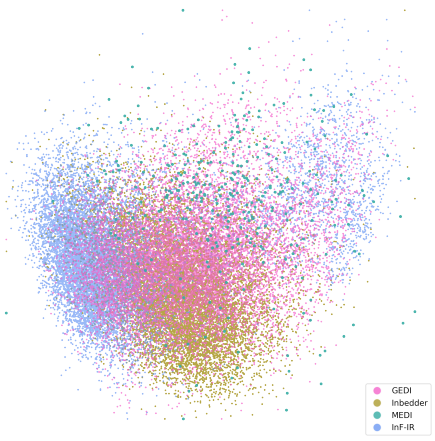

Figure 4: 2D visualization of the semantic distribution of instruction embeddings across the InF-IR, GEDI, Inbedder, and MEDI datasets using PCA.

3.2 Instruction Contrastive Evaluation (ICE)

Conventional embedding benchmarks evaluate downstream task performance but fail to assess how well embeddings reflect instruction semantics. To quantify this capability, we introduce the Instruction Contrastive Evaluation (ICE) framework. As illustrated in Figure 5, the semantic similarity of the same text can vary depending on the given instruction. For example, under the instruction *"What is the animal mentioned?"*, the sentences *"I love cat!"* and *"I dislike cat!"* should be close in the embedding space, while *"I love dog!"* should be distant. A model that does not adapt to instructions would fail this basic test. To systematically measure instruction understanding, we propose the following three metrics.

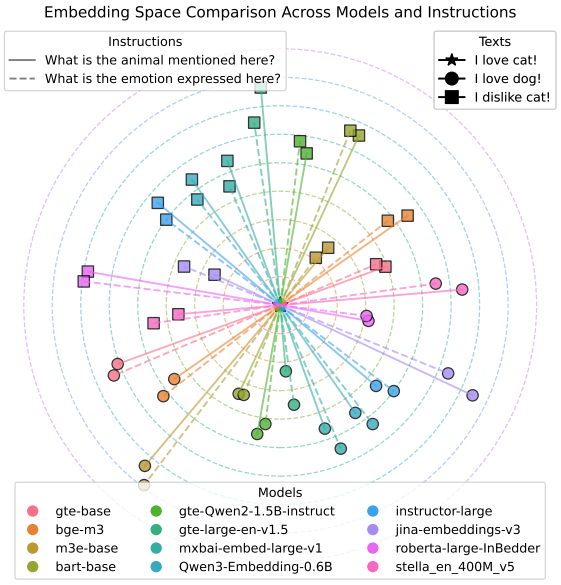

Figure 5: Embedding space comparison across instruction-aware models under varying instructions.

Instruction Retrieval Score (IRS) IRS evaluates an embedding model's ability to retrieve the correct output given an `Instruction+Input` query. We sample 3,000 triplets (I_i, Q_i, O_i) from each dataset, encoding them into a retrieval pool containing instructions, inputs, and outputs. For each query (I_i+Q_i), the model retrieves the top semantically aligned output O_i. IRS is the top-1 accuracy:

$$\text{IRS} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left[\arg\max_j \text{sim}(\mathbf{e}_{iq}^{(i)}, \mathbf{e}_{all}) = \text{id}(O_i)\right] \tag{1}$$

A high Instruction Retrieval Score (IRS) signifies the model's ability to contextually interpret instructions, perform semantic (rather than surface-level) matching, and exhibit strong discriminative retrieval capability, making it a crucial metric for evaluating instruction-aware retrieval in RAG and agent systems.

Symbol	Description
$\mathbf{e}_{iq}^{(i)}$	Embedding of $I_i + Q_i$
$\mathbf{e}_{all}$	Embedding of all candidates
$\text{id}(O_i)$	Index of gold output O_i
$\text{sim}(\cdot,\cdot)$	Similarity function
$\mathbb{1}[\cdot]$	Indicator function

Table 2: Notation for IRS.

Instruction Guidance Strength (IGS) Measures embedding models' ability to prioritize instruction semantics over surface similarity. The core requirement: `Instruction+Input` embedding should align closer to the correct `Positive` than to any `Negative`, even if the latter is lexically closer to the `Input`.

Motivation. Standard models excel at paraphrase detection but may fail when instructional intent conflicts with lexical cues. IGS evaluates instruction-aware retrieval.

$$\text{IGS} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left[\text{sim}(\mathbf{e}_{iq}^{(i)}, \mathbf{e}_{p}^{(i)}) > \text{sim}(\mathbf{e}_{iq}^{(i)}, \mathbf{e}_{n}^{(i)})\right] \tag{2}$$

Instruction Target Differential Score (ITDS) Quantifies the margin by which positives outperform negatives in embedding space, assessing alignment confidence beyond binary ranking:

$$\text{ITDS} = \frac{1}{N}\sum_{i=1}^{N}\left[\text{sim}(\mathbf{e}_{iq}^{(i)}, \mathbf{e}_{p}^{(i)}) - \frac{1}{K}\sum_{j=1}^{K}\text{sim}(\mathbf{e}_{iq}^{(i)}, \mathbf{e}_{n}^{(j)})\right] \tag{3}$$

Key properties:

ITDS employs K negative samples per instance, where higher values indicate stronger instruction separation, thereby complementing IGS by quantifying alignment strength.

3.3 Dual-Level Instructional Contrastive Learning (DICL)

We identify two dimensions of instruction understanding: *internal* (semantic meaning capture)

and *external* (intent-driven discrimination). DICL jointly optimizes both through:

The Instruction Retrieval Score (IRS) provides a dual-dimensional assessment of instruction comprehension, evaluating both the internal quality of instruction-guided semantic representation and the external sensitivity to instruction-driven output distinctions.

Notation. Let the training set be $\mathcal{D} = \{(q, p, \mathcal{N})\}$, where each query sample $q = (x_q^{\text{inst}}, x_q^{\text{text}})$ is paired with a positive sample $p = (x_p^{\text{inst}}, x_p^{\text{text}})$ and a set of negative samples $\mathcal{N} = \{(x_n^{\text{inst}}, x_n^{\text{text}})\}$.

Symbol	Description
x^{inst}	Instruction input (e.g., task prompt)
x^{text}	Content input (e.g., example or main text)
x^{mix}	Concatenation: $\texttt{concat}(x^{\text{inst}}, x^{\text{text}})$
$f(\cdot)$	Shared encoder network
$\mathbf{h} = f(x)$	Embedding of input x
$\text{sim}(\mathbf{a}, \mathbf{b})$	Cosine similarity: $\frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|}$
γ	Margin in hinge loss
τ	Temperature for InfoNCE softmax
s_{ij}	Scaled similarity: $\frac{\text{sim}(f(x_i), f(x_j))}{\tau}$
N	Batch size

Table 3: Notation used in training objectives.

1. Mixed Input Construction.

$$x^{\text{mix}} = \texttt{concat}(x^{\text{inst}}, x^{\text{text}})$$

2. Semantic Encoder and Similarity.

$$\mathbf{h} = f(x), \quad \text{sim}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|}$$

3. Improvement Loss. Encourages full inputs to outperform partial inputs:

$$\begin{aligned}\mathcal{L}_{\text{imp}} = &\max\left(0, \gamma - \text{sim}(f(x_q^{\text{mix}}), f(x_p^{\text{mix}}))\right. \\ &+ \text{sim}(f(x_q^{\text{text}}), f(x_p^{\text{mix}})) \\ &+ \max\left(0, \gamma - \text{sim}(f(x_q^{\text{mix}}), f(x_p^{\text{mix}}))\right. \\ &+ \text{sim}(f(x_q^{\text{inst}}), f(x_p^{\text{mix}}))\end{aligned}$$

4. Contrastive Loss. Distinguishes positive and negative samples:

$$\begin{aligned}\mathcal{L}_{\text{con}} = &\max\left(0, \gamma - \text{sim}(f(x_q^{\text{mix}}), f(x_p^{\text{mix}}))\right. \\ &+ \text{sim}(f(x_q^{\text{mix}}), f(x_n^{\text{mix}}))\end{aligned}$$

5. Batch-wise InfoNCE Loss. Encourages intra-batch discrimination:

$$\mathcal{L}_{\text{batch}} = \frac{1}{N} \sum_{i=1}^{N} -\log \frac{e^{s_{ii}}}{\sum_{j=1}^{N} e^{s_{ij}}},$$
$$s_{ij} = \frac{\text{sim}(f(x_i^{\text{mix}}), f(x_j^{\text{mix}}))}{\tau}$$

6. Symmetric InfoNCE Loss. Swaps anchor and target:

$$\mathcal{L}_{\text{sym}} = \frac{1}{N} \sum_{i=1}^{N} -\log \frac{e^{s_{ii}}}{\sum_{j=1}^{N} e^{s_{ji}}}$$

7. Total Objective. Final training loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{imp}} + \mathcal{L}_{\text{con}} + \mathcal{L}_{\text{batch}} + \mathcal{L}_{\text{sym}}$$

3.4 Asymmetric Fusion Distillation

Embedding model fine-tuning often suffers from catastrophic forgetting. To address this while leveraging embedding space redundancy (Yoon et al., 2024), we propose **Asymmetric Fusion Distillation (AFD)**. AFD combines embeddings from (1) a base model and (2) a new instruction-tuned model using asymmetric weighting:

$$E_{\text{fused}}(x) = \lambda \cdot E_{\text{new}}(x) + (1 - \lambda) \cdot E_{\text{base}}(x) \quad (4)$$

where $\lambda \in [0.4, 0.6]$ balances new capabilities against base knowledge retention.

A student model is trained to mimic this fused representation:

$$\mathcal{L}_{\text{AFD}} = \frac{1}{|\mathcal{N}|} \sum_{x \in \mathcal{N}} \|E_{\text{student}}(x) - E_{\text{fused}}(x)\|^2 \quad (5)$$

This distillation preserves base model robustness while incorporating new adaptations, minimizing performance degradation on original tasks.

4 Experimental

4.1 Training Configuration

The training was conducted on a single NVIDIA GeForce RTX 3090 GPU using the PyTorch framework with CUDA acceleration. Employing a parameter-efficient fine-tuning strategy, only the last 4 layers of the model were updated while all preceding layers remained frozen. The optimizer used was AdamW with a learning rate of 1e-5 and a linear warmup schedule. The model was trained for 5 epochs with a batch size of 8, and a fixed random seed of 42 was set to ensure reproducibility.

Model Name	MTEB(eng, v2)							
	Class. 11	Cluster 6	Pair. 3	Rerank 4	Retri. 7	STS 10	Sum. 1	**Avg** 42
gtr-t5-large	66.54|63.90	32.65|29.38	85.33|82.68	55.36|51.81	48.82|45.51	60.10|62.90	29.48|29.30	59.65|57.51
instructor-large	**70.42**|68.54	**37.73**|**36.26**	**86.16**|75.64	**57.44**|51.88	**50.72**|42.88	**65.62**|**65.01**	**30.36**|29.31	**63.08**|59.11
gtr-t5-large-DICL								
with MEDI	66.01|64.43	28.09|27.59	85.11|85.14	55.63|53.89	50.13|**48.20**	61.33|63.22	28.64|28.26	59.19|58.44
with InBedder	66.48|63.69	29.85|28.35	85.05|83.33	55.53|52.88	49.09|45.69	62.33|63.76	29.78|28.39	59.55|57.66
with GEDI	66.51|**68.70**	32.01|33.10	84.72|**85.51**	55.75|**55.72**	48.51|47.75	61.87|63.41	30.29|**30.82**	59.64|**60.63**
roberta-large	**62.90**|59.13	25.24|25.62	57.78|54.95	41.92|41.43	14.59|11.31	36.26|36.93	28.63|27.44	42.29|40.23
roberta-large-InBedder	61.37|**65.86**	21.55|27.78	37.18|61.17	39.84|43.26	14.18|17.60	32.99|44.59	**29.39**|27.98	39.30|47.02
roberta-large-DICL								
with MEDI	62.79|61.39	**30.96**|**33.31**	51.72|70.00	**42.86**|**49.33**	16.85|29.16	39.84|38.69	28.79|26.57	43.42|48.60
with InBedder	58.07|58.47	20.90|26.84	22.27|63.76	37.15|44.94	11.70|14.36	19.54|43.24	29.31|15.28	32.06|43.74
with GEDI	60.40|58.69	27.49|29.01	**63.52**|**72.66**	42.11|43.59	**22.32**|**30.85**	**50.66**|**51.44**	28.57|**28.48**	**47.78**|**50.37**
stella_en_400M_v5	**74.74**|**84.07**	**40.23**|41.74	**85.92**|85.43	**58.14**|**58.29**	**54.89**|**55.24**	**69.10**|70.52	**30.60**|31.66	**65.67**|**68.33**
stella_en_400M-DICL								
with MEDI	72.54|79.83	36.44|**43.06**	76.72|83.52	54.21|57.57	21.82|49.83	41.55|66.97	27.15|30.51	52.58|65.67
with InBedder	64.02|81.93	31.32|39.99	32.62|83.64	48.19|55.69	14.02|44.12	13.85|38.11	30.28|30.96	35.88|60.62
with GEDI	67.64|82.31	32.96|39.29	48.84|84.49	41.32|54.75	31.28|50.99	52.80|68.71	30.52|32.22	51.10|66.07
stella_en_400M-Distill	74.19|83.44	37.88|40.78	75.60|**86.10**	53.31|58.00	49.80|54.30	67.18|**71.66**	30.46|**32.04**	62.77|68.07

Table 4: We compare baseline models, instruction-tuned baselines (e.g., Instructor-large, roberta-large-InBedder), and our proposed DICL-based models trained on three different datasets: MEDI, InBedder, and GEDI. Models trained with DICL generally outperform their respective baselines, **with GEDI**-trained models achieving the best average performance, indicating stronger generalization across task types.

4.2 Main Result Analysis

Table 4 compares baseline models, previous instruction-tuned baselines (e.g., `Instructor-large`, `roberta-large-InBedder`), and our DICL-trained models (using MEDI, InBedder, and GEDI datasets) on the MTEB benchmark (English tasks). Key observations include:

Instruction Sensitivity: Many baseline models degrade when given an instruction. For example, `roberta-large` drops from 42.29 to 40.23 when instructions are added. This shows that the model does not adapt well to instructions. In contrast, instruction-tuned models like `roberta-large-InBedder` see mixed effects: their performance without instructions is lower, but they benefit from having instructions.

Effectiveness of DICL: Our DICL framework consistently improves instruction robustness. For `roberta-large`, the DICL/GEDI variant achieves 50.37 average (with instructions) versus 40.23 for the base, closing the gap and even exceeding base performance. Similarly, `gtr-t5-large-DICL/GEDI` improves instruction-assisted retrieval by +3.12 points (from 57.51 to 60.63) without losing base accuracy (59.65 to 59.64).

GEDI Superiority: Models trained on the GEDI dataset show the best overall performance. For instance, `roberta-large-DICL/GEDI` outperforms the MEDI-trained version by +1.77 average points (50.37 vs. 48.60), especially on retrieval (30.85 vs. 29.16) and semantic similarity tasks (51.44 vs. 38.69). GEDI's diverse instruction coverage leads to stronger generalization. While MEDI-trained models excel on some classification tasks (due to task-specific tuning), GEDI yields superior balanced gains across all task types.

Instruction Robustness: Unlike fixed-instruction models, DICL can handle unseen instructions. The `instructor-large` baseline actually regresses when presented with novel instructions (63.08 vs. 59.11), whereas our `gtr-t5-large-DICL/GEDI` improves under instruction (59.64 to 60.63). This demonstrates DICL's ability to generalize to new instructions, solving a major limitation of prior methods.

4.2.1 Evaluation on Instruction Comprehension (ICE)

Table 5 shows the ICE results on our test sets. We see that DICL-trained models dramatically outperform baselines in instruction understanding:

Instruction Retrieval Score (IRS): `roberta-large-DICL` achieves an IRS of 19.15, a +10.20 gain over the 8.95 of vanilla `roberta-large`. `gtr-t5-large-DICL` reaches 26.75 IRS, compared to 15.20 for `instructor-large`. The `stella_400M-DICL` model achieves 52.05 IRS, which is 3.74× higher than the 13.90 of the original `stella_400M`. These

huge gains show that our models can correctly retrieve outputs using the instructions as guidance.

Model Name	ICE		
	IRS	*IGS*	*ITDS*
bart-base	8.95	64.32	5.42
bge-m3	12.70	63.59	10.47
gte-base	17.05	63.10	4.96
mxbai-embed-large-v1	12.10	63.82	13.86
jina-embeddings-v3	17.60	65.10	12.52
gtr-t5-large	11.35	62.60	1.21
instructor-large	15.20	62.23	0.67
gtr-t5-large-DICL	**26.75**	**67.24**	**4.30**
roberta-large	8.95	63.41	0.016
roberta-large-InBedder	9.90	64.94	0.024
roberta-large-DICL	**19.15**	**79.27**	**6.02**
stella_en_400M_v5	13.90	64.82	4.13
stella_en_400M-DICL	**52.05**	**83.47**	**6.76**
stella_en_400M-Distill	34.35	74.44	4.66

Table 5: Evaluation of various embedding models on the proposed **Instruction Contrastive Evaluation (ICE)** metrics, including Instruction Retrieval Score (IRS) on the GEDI and MEDI subsets, Instruction Guidance Strength (IGS), and Instruction Target Differential Score (ITDS).

Instruction Guidance Strength (IGS): DICL models also boost IGS substantially (e.g., `roberta-large-DICL` has IGS 79.27 vs 63.41 for the base), indicating that they prioritize the correct (instruction-consistent) output even over distracting alternatives.

Instruction Target Differential Score (ITDS): The ITDS margin is also much higher for DICL models (e.g., 6.02 vs 0.024 for roberta variants), reflecting stronger confidence in the correct output.

Overall, the ICE benchmark confirms that our DICL approach fundamentally improves the instruction-awareness of embeddings, a capability largely missing in previous methods. In other words, we effectively teach the model *why* two outputs are equivalent or different based on the instruction.

4.2.2 Evaluation on Instruction-Following

Following recent work on instruction-following retrieval such as InF-IR, which highlights the value of fine-grained and contrastive <instruction, query, passage> supervision, we construct a small-scale but high-quality evaluation dataset using a multi-stage LLM-based generation and verification pipeline. This process yields approximately 3,000 instruction–query–document triplets with strong semantic contrast.

Model	R04	N21	C17	FIR
Models with Inf-Embed (reported from InF-IR)				
Llama-3.2-1B-Inst	-2.1	0.6	0.2	-0.4
+ Inf-Embed	5.6	3.8	1.9	3.8
Qwen2.5-3B-Inst	-1.3	2.4	-0.4	0.2
+ Inf-Embed	3.3	1.8	3.7	2.9
e5-base-v2	-6.7	-2.0	-2.9	-3.9
+ Inf-Embed	**6.9**	3.2	5.3	5.1
Models trained with GEDI-DICL (ours)				
e5-base-v2	-6.7	-2.0	-2.9	-3.9
+ GEDI-DICL	0.9	11.3	10.6	7.6
Qwen3-Emb-0.6B	2.3	2.6	6.1	3.7
+ GEDI-DICL	6.6	11.4	**13.1**	10.4
stella_en_400M_v5	-4.7	1.7	0.9	-0.7
+ GEDI-DICL	3.7	**19.2**	10.8	**11.2**

Table 6: p-MRR on Robust04 (R04), News21 (N21), Core17 (C17), and FollowIR (FIR). Inf-Embed results are reported from the original InF-IR work, while GEDI-DICL results are obtained under our training framework.

To better leverage this finer-grained supervision, we apply *Dual-Level Instructional Contrastive Learning* (DICL) with a tier-aware contrastive objective. The core learning principle of DICL remains unchanged, while higher- and lower-quality instruction–query variants are distinguished through a unified InfoNCE formulation:

$$\mathcal{L}(\mathbf{a}_i, \mathbf{p}_i) = -\log \frac{\exp\big(s(\mathbf{a}_i, \mathbf{p}_i)\big)}{\sum_{\mathbf{z}\in\mathcal{C}_i} \exp\big(s(\mathbf{a}_i, \mathbf{z})\big)}, \quad (6)$$

where $s(\cdot,\cdot)$ denotes temperature-scaled cosine similarity. We defer the full loss instantiation to Appendix A.

Table 6 presents p-MRR improvements on four instruction-following retrieval benchmarks under two different training settings. The upper block reports results from Inf-Embed, which are taken directly from the InF-IR work and trained on their curated instruction-following corpus. Consistent with prior findings, Inf-Embed yields clear gains over instruction-tuned baselines, confirming the effectiveness of fine-grained and highly contrastive <instruction, query, passage> supervision for instruction-aware retrieval.

The lower block reports results obtained under our training framework using GEDI-DICL. Across all evaluated backbones, GEDI-DICL consistently improves instruction-following performance, with

particularly strong gains on News21 and Core17. For instance, GEDI-DICL improves e5-base-v2 by +11.3 and +10.6 p-MRR on News21 and Core17, respectively, and yields consistent improvements across all datasets for Qwen3-Emb-0.6B. These results indicate that explicitly modeling hierarchical relevance among instruction–query variants enables more effective utilization of instruction-following signals than uniform contrastive training.

Notably, GEDI-DICL demonstrates robust effectiveness across diverse embedding architectures, suggesting that the proposed training objective is largely model-agnostic. The improvements are especially pronounced on datasets with more complex or implicit instructions, such as News21 and FollowIR, highlighting the benefit of structured, tier-aware contrastive learning under instruction-aware supervision.

4.3 Asymmetric Fusion Distillation Analysis

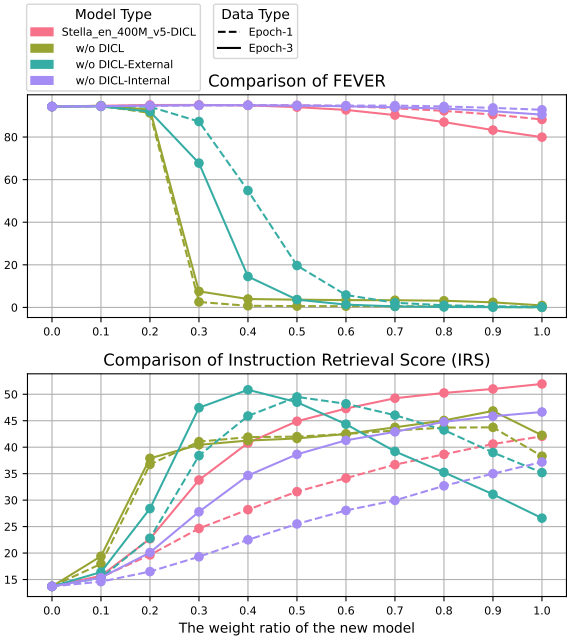

Figure 6: This figure illustrates the impact of blending the fine-tuned model with the original pre-trained model at varying weight ratios on multiple accuracy metrics. The x-axis represents the weight ratio of the new model, while the y-axis displays the corresponding performance scores across different evaluation tasks. The comparison includes our full model alongside three ablated variants.

Figure 6 analyzes how the fused model performs as we vary λ. We observe:

The full DICL model exhibits strong robustness, maintaining over 90% of base FEVER accuracy across a wide range of λ while consistently improving instruction responsiveness (IRS). In contrast, removing DICL leads to severe semantic drift, with FEVER accuracy collapsing once $\lambda > 0.3$, demonstrating the necessity of contrastive supervision. Although DICL fine-tuning alone slightly degrades general embedding quality, the final Asymmetric Fusion Distillation (AFD) stage effectively recovers it: the distilled model retains 99.6% of the original MTEB performance while preserving instruction awareness. Notably, the fusion behavior stabilizes within three epochs, indicating rapid convergence of the training pipeline.

5 Limitations

Our framework relies on automatically constructed instruction–text supervision, which may not fully capture the diversity of real-world instructions despite consistency-based filtering. In addition, we focus on single-instruction conditioning and do not explicitly address multi-instruction or compositional settings. Finally, while our method generalizes across multiple embedding backbones, its scalability to larger foundation models remains to be explored.

6 Conclusion

We present a comprehensive framework for instruction-aware text embeddings. Our contributions include a **Dual-Level Instructional Contrastive Learning (DICL)** objective, an **Asymmetric Fusion Distillation (AFD)** strategy, the **GEDI** dataset, and the **ICE** benchmark. Experiments demonstrate that our method effectively resolves the tension between instruction sensitivity and general-purpose utility, enabling a single model to follow diverse instructions without compromising its semantic capabilities. This work provides a foundation for more adaptable, user-aligned embedding models, with future directions including extension to multimodal inputs and more complex, compositional instructions.

References

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. 2022. Task-aware retrieval with instructions. *arXiv preprint arXiv:2211.09260*.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multi-linguality, multi-functionality,

multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 2318–2335.

Jiafeng Guo, Yinqiong Cai, Keping Bi, Yixing Fan, Wei Chen, Ruqing Zhang, and Xueqi Cheng. 2025. Came: Competitively learning a mixture-of-experts model for first-stage retrieval. *ACM Transactions on Information Systems*, 43(2):1–25.

Yiwei Li, Jiayi Shi, Shaoxiong Feng, Peiwen Yuan, Xinglin Wang, Boyuan Pan, Heda Wang, Yao Hu, and Kan Li. 2024. Instruction embedding: Latent representations of instructions towards task identification. volume 37, pages 87683–87711.

Wenxiao Liu, Zihong Yang, Chaozhuo Li, Zijin Hong, Jianfeng Ma, Zhiquan Liu, Litian Zhang, and Feiran Huang. 2024. Hncse: Advancing sentence embeddings via hybrid contrastive learning with hard negatives. *arXiv preprint arXiv:2411.12156*.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM transactions on audio, speech, and language processing*, 24(4):694–707.

Letian Peng, Yuwei Zhang, Zilong Wang, Jayanth Srinivasa, Gaowen Liu, Zihan Wang, and Jingbo Shang. 2024. Answer is all you need: Instruction-following text embedding via answering the question. Annual Conf. of the Association for Computational Linguistics (ACL) 2024.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A Smith, Luke Zettlemoyer, and Tao Yu. 2023. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121.

Julien Subercaze, Christophe Gravier, and Frédérique Laforest. 2015. On metric embedding for boosting semantic similarity computations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 8–14.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916.

Jinsung Yoon, Rajarishi Sinha, Sercan Arik, and Tomas Pfister. 2024. Matryoshka-adaptor: Unsupervised and supervised tuning for smaller embedding dimensions. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10318–10336.

Peitian Zhang, Shitao Xiao, Zheng Liu, Zhicheng Dou, and Jian-Yun Nie. 2023. Retrieve anything to augment large language models. *CoRR*.

Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models. In *The Thirteenth International Conference on Learning Representations*.

Yuchen Zhuang, Aaron Trinh, Rushi Qiang, Haotian Sun, Chao Zhang, Hanjun Dai, and Bo Dai. 2025. Towards better instruction following retrieval models. *arXiv preprint arXiv:2505.21439*.

A Tiered Ranking Contrastive Loss

A.1 Task Formulation

Let $\{\mathcal{D}_i\}_{i=1}^{B}$ denote a mini-batch of training instances with batch size B. Each instance $\mathcal{D}_i$ consists of: (i) a ground-truth target text p_i, (ii) a positive instruction I_i^+ and a positive query Q_i^+, (iii) a negative instruction I_i^- and a negative query Q_i^-, and (iv) a set of external negative texts $\{n_{i,k}\}_{k=1}^{K_i}$.

All texts are encoded by a shared text encoder and ℓ_2-normalized to obtain vector representations. We denote the embedding of a text x as $\mathbf{e}_x$.

To model instruction–query interactions, we additionally construct composed representations

$$\mathbf{e}_{I_i^+Q_i^+} = f(I_i^+, Q_i^+), \tag{7}$$

$$\mathbf{e}_{I_i^+Q_i^-} = f(I_i^+, Q_i^-), \tag{8}$$

$$\mathbf{e}_{I_i^-Q_i^+} = f(I_i^-, Q_i^+), \tag{9}$$

where $f(\cdot)$ concatenates the instruction and query into a single input sequence before encoding.

A.2 Tiered Relevance Structure

We categorize candidate representations into five relevance tiers according to their semantic quality with respect to the ground-truth target:

- **Tier 1**: composed positive instruction and query (I^+, Q^+),
- **Tier 2**: single positive instruction I^+ or single positive query Q^+,
- **Tier 3**: partially corrupted compositions (I^+, Q^-) or (I^-, Q^+),
- **Tier 4**: single negative instruction I^- or negative query Q^-,
- **Tier 5**: external negative texts.

Our objective is to learn an embedding space in which representations from higher tiers are consistently ranked above those from lower tiers.

A.3 Similarity Function

We employ a temperature-scaled cosine similarity:

$$s(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^\top \mathbf{b}}{\tau}, \tag{10}$$

where τ is a temperature hyperparameter.

A.4 Tier-1 Ranking Loss with Target Anchors

We first treat the ground-truth target embedding $\mathbf{e}_{p_i}$ as the anchor and enforce the Tier-1 representation to be the most similar. The loss is defined as an InfoNCE-style classification objective:

$$\mathcal{L}_{\text{T1}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp\big(s(\mathbf{e}_{p_i}, \mathbf{e}_{I_i^+Q_i^+})\big)}{\sum_{\mathbf{z}\in\mathcal{C}_i^{(1)}} \exp\big(s(\mathbf{e}_{p_i}, \mathbf{z})\big)}, \tag{11}$$

where the candidate set $\mathcal{C}_i^{(1)}$ includes: all Tier-2, Tier-3, Tier-4, Tier-5 representations of instance i, as well as Tier-1 representations from other instances in the same mini-batch, which serve as hard negatives.

A.5 Symmetric Target–Tier Alignment

To encourage symmetric alignment, we additionally reverse the anchor and positive roles by treating the Tier-1 representation as the anchor:

$$\mathcal{L}_{\text{sym}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp\big(s(\mathbf{e}_{I_i^+Q_i^+}, \mathbf{e}_{p_i})\big)}{\sum_{\mathbf{z}\in\mathcal{C}_i^{(2)}} \exp\big(s(\mathbf{e}_{I_i^+Q_i^+}, \mathbf{z})\big)}. \tag{12}$$

Here, $\mathcal{C}_i^{(2)}$ consists of external negatives and target embeddings from other batch instances.

A.6 Tier-2 Anchor Ranking Loss

We further constrain single positive components to be closer to the target than lower-quality tiers. Specifically, we define two losses using I_i^+ and Q_i^+ as anchors:

$$\mathcal{L}_{I^+} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp\big(s(\mathbf{e}_{I_i^+}, \mathbf{e}_{p_i})\big)}{\sum_{\mathbf{z}\in\mathcal{C}_i^{(3)}} \exp\big(s(\mathbf{e}_{I_i^+}, \mathbf{z})\big)}, \tag{13}$$

$$\mathcal{L}_{Q^+} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp\big(s(\mathbf{e}_{Q_i^+}, \mathbf{e}_{p_i})\big)}{\sum_{\mathbf{z}\in\mathcal{C}_i^{(4)}} \exp\big(s(\mathbf{e}_{Q_i^+}, \mathbf{z})\big)}. \tag{14}$$

In both cases, the negative candidate sets include Tier-3 and Tier-4 representations, external negatives, and target embeddings from other batch instances.

A.7 Explicit Tier-1 vs. Tier-2 Margin Enforcement

To explicitly enforce that Tier-1 representations dominate Tier-2, we introduce an additional contrastive loss:

$$\mathcal{L}_{\text{T1}>\text{T2}} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\exp\left(s(\mathbf{e}_{p_i}, \mathbf{e}_{I_i^+Q_i^+})\right)}{\sum_{\mathbf{z}\in\mathcal{C}_i^{(2)}}\exp\left(s(\mathbf{e}_{p_i}, \mathbf{z})\right)}, \tag{15}$$

where $\mathcal{C}_i^{(2)}$ consists of the Tier-1 positive and all Tier-2 candidates.

A.8 Overall Objective

The final training objective is the sum of all loss terms:

$$\mathcal{L} = \mathcal{L}_{\text{T1}} + \mathcal{L}_{\text{sym}} + \mathcal{L}_{I^+} + \mathcal{L}_{Q^+} + \mathcal{L}_{\text{T1}>\text{T2}}. \tag{16}$$

This tiered ranking formulation enables fine-grained supervision over instruction–query compositions, encouraging the model to capture hierarchical relevance relationships beyond binary positive–negative distinctions.

Table 9: Different Task Instruction

Task Name	**Instruction**
ArguAna	Query Instruction: Represent the argument for retrieving counter-arguments: Doc Instruction: Represent the counter-argument for retrieval:
CQADupstackAndroidRetrieval	Query Instruction: Represent the Android-related question for retrieving supporting answers: Doc Instruction: Represent the Android-related answer for retrieval:
CQADupstackEnglishRetrieval	Query Instruction: Represent the English language question for retrieving supporting answers: Doc Instruction: Represent the English language answer for retrieval:
CQADupstackGamingRetrieval	Query Instruction: Represent the gaming-related question for retrieving supporting answers: Doc Instruction: Represent the gaming-related answer for retrieval:
CQADupstackGisRetrieval	Query Instruction: Represent the GIS-related question for retrieving supporting answers: Doc Instruction: Represent the GIS-related answer for retrieval:
CQADupstackMathematicaRetrieval	Query Instruction: Represent the Mathematica-related question for retrieving supporting answers: Doc Instruction: Represent the Mathematica-related answer for retrieval:
CQADupstackPhysicsRetrieval	Query Instruction: Represent the physics-related question for retrieving supporting answers: Doc Instruction: Represent the physics-related answer for retrieval:
CQADupstackProgrammersRetrieval	Query Instruction: Represent the programming question for retrieving supporting answers: Doc Instruction: Represent the programming answer for retrieval:
CQADupstackStatsRetrieval	Query Instruction: Represent the statistics question for retrieving supporting answers: Doc Instruction: Represent the statistics answer for retrieval:
CQADupstackTexRetrieval	Query Instruction: Represent the TeX/LaTeX question for retrieving supporting answers: Doc Instruction: Represent the TeX/LaTeX answer for retrieval:
CQADupstackUnixRetrieval	Query Instruction: Represent the Unix-related question for retrieving supporting answers: Doc Instruction: Represent the Unix-related answer for retrieval:
CQADupstackWebmastersRetrieval	Query Instruction: Represent the webmaster-related question for retrieving supporting answers: Doc Instruction: Represent the webmaster-related answer for retrieval:
CQADupstackWordpressRetrieval	Query Instruction: Represent the WordPress-related question for retrieving supporting answers: Doc Instruction: Represent the WordPress-related answer for retrieval:
ClimateFEVER	Query Instruction: Represent the climate-related claim for retrieving supporting evidence: Doc Instruction: Represent the climate-related evidence for retrieval:
DBPedia	Query Instruction: Represent the question about general knowledge for retrieving supporting documents: Doc Instruction: Represent the general knowledge document for retrieval:
FEVER	Query Instruction: Represent the factual claim for retrieving supporting evidence: Doc Instruction: Represent the factual evidence for retrieval:

Task Name	Instruction
HotpotQA	Query Instruction: Represent the multi-hop question for retrieving supporting facts: Doc Instruction: Represent the factual information for retrieval:
NFCorpus	Query Instruction: Represent the medical question for retrieving supporting documents: Doc Instruction: Represent the medical document for retrieval:
NQ	Query Instruction: Represent the Wikipedia question for retrieving supporting documents: Doc Instruction: Represent the Wikipedia document for retrieval:
SCIDOCS	Query Instruction: Represent the scientific question for retrieving supporting papers: Doc Instruction: Represent the scientific paper for retrieval:
SciFact	Query Instruction: Represent the scientific claim for retrieving supporting evidence: Doc Instruction: Represent the scientific evidence for retrieval:
TRECCOVID	Query Instruction: Represent the COVID-19 related question for retrieving supporting documents: Doc Instruction: Represent the COVID-19 related document for retrieval:
Touche2020	Query Instruction: Represent the argumentative question for retrieving supporting arguments: Doc Instruction: Represent the argument for retrieval:
MSMARCO	Query Instruction: Represent the web search query for retrieving relevant documents: Doc Instruction: Represent the web document for retrieval:
AmazonCounterfactualClassification	Instruction: Represent the text for determining if it contains counterfactual statements:
AmazonPolarityClassification	Instruction: Represent the product review for classifying sentiment as positive or negative:
AmazonReviewsClassification	Instruction: Represent the review text for categorizing its overall sentiment or rating:
Banking77Classification	Instruction: Represent the customer query for classifying its intent in banking services:
EmotionClassification	Instruction: Represent the sentence for classifying the expressed emotion (e.g., joy, sadness, anger, fear):
ImdbClassification	Instruction: Represent the movie review for classifying sentiment as positive or negative:
MTOPDomainClassification	Instruction: Represent the utterance for classifying its domain in task-oriented dialogue:
MTOPIntentClassification	Instruction: Represent the utterance for classifying its intent in task-oriented dialogue:
MassiveIntentClassification	Instruction: Represent the utterance for classifying its intent in multilingual task-oriented dialogue:
MassiveScenarioClassification	Instruction: Represent the utterance for classifying its scenario in multilingual task-oriented dialogue:
ToxicConversationsClassification	Instruction: Represent the conversation text for detecting toxic content:
TweetSentimentExtractionClassification	Instruction: Represent the tweet for extracting and classifying its sentiment:
ArxivClusteringS2S	Instruction: Represent the Arxiv paper for set-to-set clustering:
BiorxivClusteringS2S	Instruction: Represent the Biorxiv paper for set-to-set clustering:
MedrxivClusteringP2P	Instruction: Represent the Medrxiv paper for pairwise clustering:
MedrxivClusteringS2S	Instruction: Represent the Medrxiv paper for set-to-set clustering:
RedditClustering	Instruction: Represent the Reddit post for clustering:
RedditClusteringP2P	Instruction: Represent the Reddit post for pairwise clustering:
StackExchangeClustering	Instruction: Represent the StackExchange post for clustering:
StackExchangeClusteringP2P	Instruction: Represent the StackExchange post for pairwise clustering:
TwentyNewsgroupsClustering	Instruction: Represent the Twenty Newsgroups post for clustering:
SprintDuplicateQuestions	Instruction: Represent the question for retrieving duplicate questions:
TwitterSemEval2015	Instruction: Represent the Tweet post for retrieving duplicate comments:
TwitterURLCorpus	Instruction: Represent the Tweet for retrieving similar URLs:
AskUbuntuDupQuestions	Query Instruction: Represent the Ubuntu-related question for retrieving duplicate questions: Doc Instruction: Represent the Ubuntu-related question for duplicate detection:
MindSmallReranking	Query Instruction: Represent the News query for retrieving articles: Doc Instruction: Represent the News article for retrieval:
SciDocsRR	Query Instruction: Represent the scientific document query for retrieving related papers: Doc Instruction: Represent the scientific document for retrieval:
StackOverflowDupQuestions	Query Instruction: Represent the programming question for retrieving duplicate questions: Doc Instruction: Represent the programming question for duplicate detection:
BIOSSES	Instruction: Represent the biomedical sentence:
SICK-R	Instruction: Represent the sentence for relatedness analysis:
STS12	Instruction: Represent the statement:
STS13	Instruction: Represent the statement:
STS14	Instruction: Represent the statement:
STS15	Instruction: Represent the statement:

Task Name	Instruction
STS16	Instruction: Represent the statement:
STS17	Instruction: Represent the statement:
STS22	Instruction: Represent the statement:
STSBenchmark	Instruction: Represent the statement for semantic textual similarity:
SummEval	Instruction: Represent the Biomedical summary for retrieving duplicate summaries:

Multi-Stage Instruction-Enhanced Data Generation Prompts
Stage 1 Prompt: Instruction–Query–Follow Framework Generation
Motivation: Construct a high-level semantic control framework to explicitly model how *Instruction* alters the interpretation of *Query* in retrieval-based vector models.
The key observation is that in instruction-aware retrieval, the Query alone is insufficient to determine the correct document; only the combination of Instruction + Query defines the retrieval intent.
Given a seed text (used only as inspiration rather than content), the model is asked to generate a **retrieval framework**, not concrete documents.
This stage explicitly separates semantic roles before any document text is produced.
The model is instructed to generate:
Instruction: A retrieval-oriented instruction describing how to interpret and process the Query.
Query: A standalone retrieval query derived from the theme of the seed text.
Pos_follow: A semantic description specifying what properties the positive document must satisfy under Instruction + Query.
Negs_follow: Multiple descriptions specifying different semantic deviation strategies for negative documents.
The output is required to be in JSON format and must not contain any actual document content.
This stage enforces that Instruction semantically governs Query interpretation and defines retrieval boundaries.
Stage 2 Prompt: Positive and Negative Document Generation
Motivation: Explicitly materialize how Instruction reshapes semantic similarity by generating documents that are indistinguishable at surface level but separable under Instruction-aware embeddings.
Using the framework produced in Stage 1, the model is instructed to generate:
Pos: A complete document that strictly follows the semantic constraints defined by Instruction + Query and Pos_follow.
Negs: Multiple complete documents that are highly similar to Pos in topic, length, and style, but violate Instruction constraints in different ways as specified by each Negs_follow.
The prompt emphasizes that all Pos and Negs are documents assumed to be pre-stored in the retrieval database, not answers to the Query.
Each negative example must fail for a distinct reason (e.g., wrong focus, missing key constraint, semantic inversion, partial relevance).
This stage operationalizes the abstract semantic rules from Stage 1 into concrete retrieval candidates, ensuring that only Instruction-aware representations can reliably distinguish Pos from Negs.
Stage 3 Prompt: Instruction Effect Explanation and Semantic Justification
Motivation: Provide an explicit semantic explanation of how Instruction modifies similarity relationships, serving both as supervision signal and interpretability anchor.
The model is asked to generate a concise explanation describing:
How the Instruction changes the semantic interpretation of the Query.
Why the Pos document satisfies the retrieval intent only under Instruction + Query.
How different Negs reflect distinct failure modes with respect to the Instruction.
This stage enforces a meta-level understanding:
the semantic similarity between (Instruction + Query) and Pos should be higher than that between Query and Pos alone.
The explanation serves as a consistency check that the generated data aligns with instruction-conditioned embedding training objectives, rather than surface-level lexical similarity.

Table 7: Prompt templates for a three-stage instruction-enhanced data generation pipeline, explicitly modeling how Instructions reshape semantic similarity in retrieval-oriented vector models.

Stage 4 Prompt: LLM-based Instruction-aware Retrieval Validation

Motivation: Use a large language model as a semantic oracle to verify whether the generated retrieval data truly reflects instruction-conditioned semantic understanding, rather than surface-level relevance.

The key idea of this stage is to transform the retrieval task into a multiple-choice reasoning problem, where the LLM must explicitly decide which candidate document best satisfies the combined semantics of *Instruction + Query*.

Given a validated retrieval instance containing one positive document (Pos) and multiple hard negative documents (Negs), the model is prompted with:

Instruction: The retrieval instruction that specifies how the query should be interpreted.

Query: The user query whose intent is conditioned by the Instruction.

Options: A shuffled list of documents, including exactly one Pos and several Negs, presented as multiple-choice candidates.

The prompt asks the LLM to select the single most appropriate option according to Instruction + Query, and to output only the option label (e.g., A, B, C).

To rigorously test instruction sensitivity, the same multiple-choice prompt is evaluated under four semantic configurations:

1) Instruction_positive + Query_positive (the model **should** select Pos).

2) Instruction_positive + Query_negative (the model **should not** select Pos).

3) Instruction_negative + Query_positive (the model **should not** select Pos).

4) Instruction_negative + Query_negative (the model **should not** select Pos).

Each configuration is evaluated multiple times with randomized option ordering to reduce positional bias.

A data instance is retained only if the LLM's selections are consistent with the expected outcomes across rounds (strict or majority-vote mode).

This stage filters out retrieval samples where the positive document can be identified without relying on instruction semantics, ensuring that the remaining data enforces instruction-aware similarity in embedding space.

Table 8: Prompt template for Stage 4: LLM-based post-verification of instruction-conditioned retrieval data using multiple-choice semantic judgment.

Paraphrase Prompt

Motivation: Vector Model instruction semantic understanding

The main phenomenon is that the existing embedding models trained on HotAQ data can understand the semantics of a text, but they are very straightforward and understand based on the text content contained in the text.

Let me give you an example

Its directive requires text

Instruction = "Please output the opposite semantic representation of the following sentence:"

If I type in the text

Input = "Apples are red"

The target text is consistent with human understanding

Output = "The apple is not red"

Then, in theory, the semantic vector obtained by (Instruction + Input) should be the semantic vector of Output, and the similarity is very high. But in fact, the semantic vector obtained by (Instruction + Input) contains the information of the Instruction text, which reduces the similarity between instruction and Output.

According to human understanding, the semantic similarity between Instruction and Output should be greater than that between Input and Output.

Of course, Instruction does not have to be in this form only; it can also be other types of instruction information.

Now please help me to judge whether the Instruction, Input and Output in the following data are consistent with the data used to train the semantic understanding of the vector model instructions. Please explain the reasons for your judgment in the way of chain of thought.

“‘json

{example_data}

“‘

Output in Json format:

“‘json

{

"CoT": ,

"Judgment": Yes or No,

}

“‘

Table 10: Prompt template for evaluating whether given Instruction-Input-Output examples align with the vector model’s instruction-based semantic understanding training paradigm.

Paraphrase Prompt

This study aims to improve the deep semantic understanding ability of vector models for complex instructions and break through the limitation of traditional text embedding models that only focus on literal matching. Current models trained on HotpotQA and other datasets can effectively capture explicit semantic features, but when faced with instructions that need to perform logical transformations (such as semantic reversal and emotional transformation), the generated semantic representations often lack instruction compliance.

Taking the task of "generating inverse semantics" as an example, the existing model is easy to identify "apples are red" and "bananas are red" as similar semantics (sharing color features), but cannot accurately identify "apples are not red" as the true semantic inversion. This indicates that the current model fails to effectively distinguish the difference between surface features and deep semantic logic when understanding operational instructions.

The research goal is to construct high-quality contrastive learning triples (Input-Positive-Negative), focusing on the following problems:

Instruction-aware modeling: Enabling models to not only understand text content, but also accurately capture the semantic requirements of manipulation instructions (e.g., negation, analogy, inference)

Semantic boundary division: By designing challenging negative examples (including synonym interference, logical conflict samples, irrelevant topic samples, etc.), the model's ability to distinguish subtle semantic differences is enhanced

Generalization ability improvement: An enhanced dataset covering multiple instruction types (negation, hypothesis, condition judgment, etc.) is constructed to adapt the model to the requirements of open domain instruction semantic understanding

Finally, it is expected to produce an instruction-sensitive semantic encoding model, whose core breakthrough lies in the ability to jointly encode operation instructions and text content, forming a structured distribution in the semantic representation space that conforms to human logic reasoning. This will provide more powerful base model support for application scenarios that require complex semantic understanding such as dialogue systems and intelligent retrieval.

Now I have the standard data for the semantic understanding of input and positive vector model instructions, but there is still a lack of Negative data to enhance the comparative learning effect of model training.

—

Input:

“‘json

{example_data}

“‘

—

Requirements:

1. The format of the negative data should be close to the length of the Positive data.

2. Generate the current negative data and the fit score of the input.

3. Score on a scale of 0-5, where 0-1 indicates information completely irrelevant to the current discussion, 2-4 indicates some relevance but error, and 5 indicates a complete correspondence (a positive score).

Think about as many different types of Negative data as possible, and gradually increase the score so that the model understands multiple dimensions.

5. Output as Json:

“‘json

{

"negative": ,

"score":

}

“‘

Table 11: Prompt template for generating instruction-aware negative samples and semantic fit scores to enhance contrastive learning in vector-based semantic models.