ASEMBED: TRAINING-FREE EMBEDDING EXTRACTION FOR ORDER-ROBUST RETRIEVAL VIA INTERNAL LLM SIGNALS

Anonymous authors
Paper under double-blind review

ABSTRACT

Training-free embedding methods reuse frozen decoder-only language models for retrieval without parameter updates. However, query content reordering can alter retrieval rankings and degrade retrieval performance. We observe that local attention concentrations remain visible under reordering, while their strength varies across layers. Based on these observations and prior findings linking attention-sink formation to representational compression, we propose ASEmbed, a training-free framework that uses *Attention Sink and Projected Value* signals to mitigate retrieval degradation under query content reordering. The framework comprises Boundary and Layer. Guided by these signals, BOUNDARY seeks to uncover intrinsic interaction patterns that persist under reordering and uses them to partition each input into contiguous blocks. AS-Attention enables bidirectional attention across these blocks while retaining direct causal attention within each block. LAYER traces the evolution of the same signals across layers to determine when to enable bidirectional cross-block interaction, preserving contextual information in earlier layers. It then selects readout layers from the representations computed with AS-Attention. Extensive experiments on three retrieval benchmarks and two reordering benchmarks demonstrate that ASEmbed achieves average nDCG@10 improvements of 1.96 and 4.28 points over the best-performing baseline on retrieval and reordering retrieval tasks, respectively.

1 INTRODUCTION

Retrieval embeddings support semantic search, code retrieval, and retrieval-augmented generation by enabling efficient retrieval over large candidate collections (Zhao et al., 2024; Xie et al., 2023; Karpukhin et al., 2020; Lewis et al., 2020). Large language models (LLMs) serve as embedding backbones with broad linguistic, semantic, and code capabilities (Tao et al., 2025; Rozière et al., 2023). Recent studies adapt LLMs into competitive embedding models through contrastive objectives and architectural modification (Wang et al., 2024; Lee et al., 2025). However, this adaptation requires task-specific data and backbone-specific tuning (Wang et al., 2024; BehnamGhader et al., 2024; Qin et al., 2025), limiting scalability across backbones.

Training-free embedding extraction offers an alternative by deriving embeddings from frozen LLMs (Tao et al., 2025; Fu et al., 2025; Tang & Yang, 2026). Existing methods can be grouped into three categories: (1) Input-side methods reshape context through prompts, sequence transformations, or generated views (Jiang et al., 2024; Springer et al., 2025; Lin et al., 2026; Thirukovalluru & Dhingra, 2025). (2) In-model methods change token interactions through attention modifications or internal states re-routing (Fu et al., 2025; Lin et al., 2025; Tang & Yang, 2026). (3) Output-side methods change the selection and aggregation of internal states across tokens and layers (Skean et al., 2025; Wang et al., 2026; Zhang et al., 2026). Together, these studies highlight how context access and the selection and aggregation of internal representations affect embedding quality.

In this work, query content reordering as rearranging the content within a query(Figure 1(a)). Figure 1(b) illustrates the changes in reciprocal rank (RR) for three representative methods (ReverseEOL (Lin et al., 2026), KV-Embedding (Tang & Yang, 2026), and Value Aggregation (VA) (Zhang et al., 2026)) under query content ordering. RR changes vary across queries for all

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.

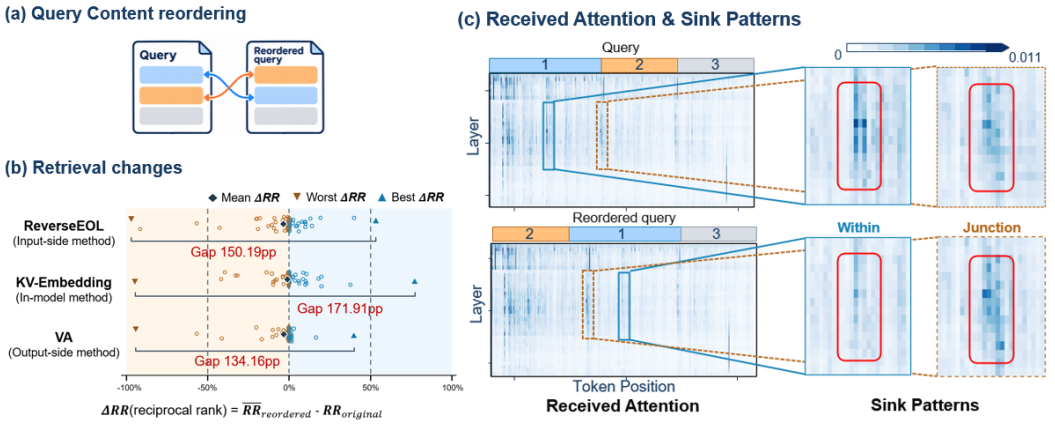

Figure 1: Content reordering affects retrieval and attention. (a) Query content reordering. (b) Changes in reciprocal rank. Gap denotes the range of ΔRR across queries in percentage points (pp). (c) Received attention for two orders of one query in Qwen3-Embedding-8B

methods, with decreases exceeding 50 percentage points in some cases. One possible explanation is that reordering changes relative positions and the preceding context accessible under causal attention, which may affect retrieval results (Springer et al., 2025; Wang et al., 2025). To mitigate order dependency, related approaches modify attention and positional assignments for predefined input segments, as Set-Based Prompting (McIlroy-Young et al., 2024) for answer options and PINE (Wang et al., 2025) for retrieved documents. For embedding extraction, we investigate how internal LLM signals can determine input partitions and guide attention modification and representation readout.

To identify such signals, we compare attention received across tokens and layers for two orders of the same query (Figure 1(c)) and make two observations. (1) In this example, the overall attention distribution changes, but local attention concentrations remain visible within the reordered segments and at their junctions, forming local *Attention Sinks* (Xiao et al., 2024; Gu et al., 2025). (2) The attention weights received at these positions vary across layers. Prior work relates attention sinks to representation formation, suggesting that early layers integrate contextual information while attention changes in middle layers accompany representational compression (Queipo-de Llano et al., 2026). These observations motivate using local attention concentrations to guide input partitioning and their changes across layers to guide attention switching. However, some sinks receive high attention weights but have value vectors with small norms, allowing attention heads to act as approximate no-ops (Fesser et al., 2026; Barbero et al., 2025). Attention weights alone therefore do not determine a token's contribution to the attention output. To identify sinks that contribute information to subsequent token representations, we therefore introduce *Projected Value* as a complementary internal signal (Kobayashi et al., 2020; Zhang et al., 2026).

Based on these observations, we propose ASEmbed, a training-free framework that uses *Attention Sink and Projected Value* (ASPV) to mitigate the decline in retrieval performance caused by query content reordering. The framework comprises BOUNDARY for input partitioning and LAYER for attention switching and readout layer selection. BOUNDARY uses ASPV signals to select block endpoints during the first inference under native causal attention. The AS-Attention uses these blocks to retain direct causal attention within blocks and enable bidirectional attention across blocks. Layer uses changes in ASPV signals across layers to select where AS-Attention begins; earlier layers retain native causal attention. It then selects readout layers using the geometry and similarity of Value representations computed during the second inference with AS-Attention.

We evaluate ASEmbed on eight natural-language and code retrieval tasks from MMTEB, CoIR, and CoREB, and assess robustness to content reordering on two datasets constructed from XCodeEval and OpenClassGen. Across the eight tasks, ASEmbed improves MRR and nDCG@10 by an average of 1.72 and 1.96 percentage points, respectively, relative to the best-performing baseline. On the two reordering datasets, its average nDCG@10 exceeds the best-performing baseline by 4.28 percentage points. Combining ASEmbed with Echo, ReverseEOL, or KV-Embedding also improves each baseline's average retrieval performance. These results support the retrieval effectiveness of ASEmbed on the original and the reordered evaluated.

The main contributions are as follows:

- We characterizes the effects of content reordering on training-free embedding extraction from decoder-only LLMs and examine their relationship with internal representation formation.
- We propose ASEmbed, a training-free framework for frozen decoder-only LLMs that determines input partitions and the attention switching layer using ASPV, then selects readout layers from the resulting representations.
- Experiments on eight retrieval tasks and two content reordering datasets evaluate the effectiveness and robustness of ASEmbed and demonstrate its compatibility with existing extraction methods.

2 Related Work

Training-Free Decoder-Only Embeddings Training-free methods extract embeddings from frozen LLMs by modifying the input, aggregating internal representations, or changing the attention mechanism. (1) Input-side methods use prompts to guide information aggregation or repeat the input to provide tokens with broader context (Jiang et al., 2024; Zhuang et al., 2024; Springer et al., 2025). Other methods further vary the order or expression of the same content and aggregate resulting representations (Lin et al., 2026; Thirukovalluru & Dhingra, 2025). (2) In-model methods modify attention or the internal states available to each token. Bidirectional attention enables later-to-earlier information flow, while summary prepending and KV rerouting make sequence-level context accessible to earlier positions (Lin et al., 2025; Fu et al., 2025; Ding et al., 2026; Tang & Yang, 2026). (3) Output-side methods extract and aggregate internal representations, including value vectors and intermediate-layer hidden states (Zhang et al., 2026). Inspired by this, ASEmbed adopts unified ASPV signals to determine input partitions and the attention switching layer.

Order Sensitivity and Structured Attention Reordering affects token representations through causal visibility and position-dependent attention computation (Xiang et al., 2024; Wang et al., 2025). Related structural approaches distinguish within-unit from cross-unit interactions. Window- and subsequence-based designs restrict direct cross-unit attention and reuse position indices (Ratner et al., 2023; McIlroy-Young et al., 2024). PINE combines cross-document bidirectional access with within-document causal visibility and relative-position reassignment (Wang et al., 2025). RoToR uses a shared global order and circular position assignment to organize interactions among segments (Yoon et al., 2025). These approaches configure attention and positional assignments over supplied partitions. ASEmbed instead induces interaction blocks from each input's ASPV signals, making the partition itself part of the extraction design.

Attention Sinks and Representation Depth The usefulness of internal representations for embedding extraction varies across model depth. Prior work finds that intermediate layers can outperform the final layer on multiple embedding tasks. Geometric analyses further identify phase-like changes in intrinsic dimension across depth (Skean et al., 2025; Cheng et al., 2025). Related work connects the layerwise evolution of attention sinks to representation compression and describes a progression through mixing, compression, and refinement (Queipo-de Llano et al., 2026). These findings point to stage-dependent representation formation across model depth. This depth dependence motivates selecting the attention switching layer separately from the depth at which retrieval representations are read. ASEmbed therefore uses layerwise ASPV signals to select the attention switching layer and the geometry of representations from the second inference to select readout layers.

3 Method

3.1 Preliminaries

Let F_θ be a frozen decoder-only LLM with L layers, and let $x = (x_1, \ldots, x_n)$ be an input sequence of n valid tokens. We denote the embedding extracted from x by $\mathbf{e}(x)$. Queries q and candidates c are encoded independently, with similarity $\mathrm{sim}(q, c) = \mathbf{e}(q)^\top \mathbf{e}(c)$. For attention head h at layer ℓ,

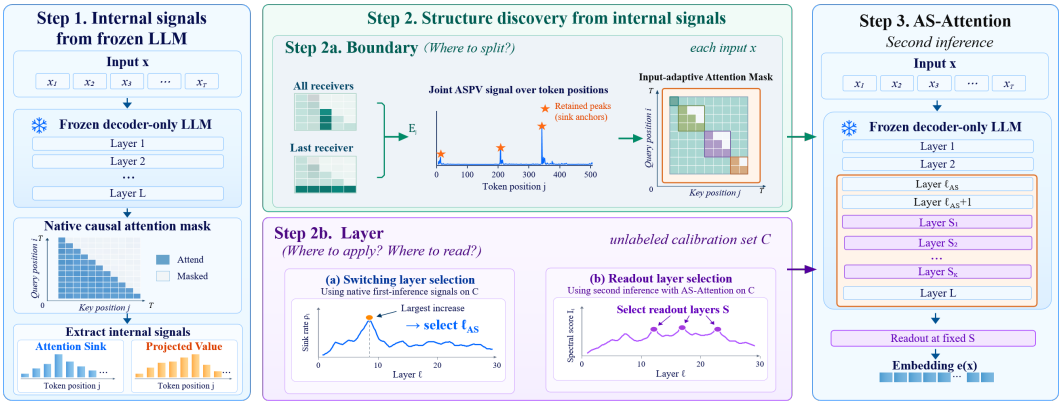

Figure 2: Overview of ASEmbed. (1) Native causal inference with a frozen decoder-only LLM provides *Attention Sink and Projected Value* (ASPV) signals. (2a) BOUNDARY uses these signals to form input-dependent partitions. (2b) On an unlabeled calibration set, LAYER selects the attention switching layer from the first inference, then selects readout layers from representations obtained during the second inference with AS-Attention. (3) Each input's second inference applies AS-Attention from the fixed switching layer onward, retaining direct causal attention within blocks and enabling bidirectional attention across blocks. Representations from the fixed readout layers are used to form the final embedding.

$A_{i,j}^{\ell,h}$ denotes the attention weight from token i to token j, and $\mathbf{z}_j^{\ell,h}$ denotes the value vector of token j after output projection. These quantities are computed as

$$A_{i,j}^{\ell,h} = \mathrm{softmax}_j\left(\frac{(\mathbf{q}_i^{\ell,h})^\top \mathbf{k}_j^{\ell,g(h)}}{\sqrt{d_h}} + M_{i,j}\right), \qquad \mathbf{z}_j^{\ell,h} = W_O^{\ell,h}\mathbf{v}_j^{\ell,g(h)}. \tag{1}$$

Here, $\mathbf{q}_i^{\ell,h}$ and $\mathbf{k}_j^{\ell,g(h)}$ are the query and key vectors after the model's native positional transformations, and $\mathbf{v}_j^{\ell,g(h)}$ is the value vector before output projection. The head dimension is d_h, and $g(h)$ identifies the key/value head associated with query head h. The matrix $W_O^{\ell,h}$ is the output projection block for head h. Under native causal attention, $M_{i,j} = 0$ for $j \le i$ and $M_{i,j} = -\infty$ otherwise.

3.2 OVERVIEW

Under causal attention, content reordering can alter retrieval representations by changing the preceding context available to tokens, while retrieval relevance is preserved. *Attention sinks* are source positions that receive concentrated attention (Xiao et al., 2024). Figure 1(c) shows local sinks in both query orders, with attention weights that vary across layers. However, sinks at the beginning of a sequence can act as approximate no-ops (Fesser et al., 2026; Barbero et al., 2025). Attention weights alone therefore do not fully capture a source's contribution to the attention output. We complement attention weights with *Projected Value*, which are value vectors after output projection. Within each head, attention weights and projected-value norms jointly determine the magnitude of this contribution (Kobayashi et al., 2020; Zhang et al., 2026).

To improve retrieval effectiveness and robustness to content reordering, we propose ASEmbed (Figure 2), a training-free framework guided by *Attention Sink and Projected Value* (ASPV) signals. Its *AS-Attention* structure enables bidirectional context access across blocks while retaining direct causal attention within each block. (1) BOUNDARY uses ASPV signals to select candidate sinks and places boundaries after them to form blocks. (2) LAYER uses changes in ASPV signals across layers to select where AS-Attention begins, preserving native causal computation in earlier layers. It then selects readout layers using the geometry and cross-layer similarity of Value representations computed with AS-Attention.

3.3 BOUNDARY: INPUT-ADAPTIVE AS-ATTENTION

BOUNDARY uses local attention sinks as candidate block endpoints, aiming to identify positions that contribute information to subsequent token representations. Under causal attention, represen-

tations at these positions can incorporate preceding context (Fu et al., 2025; Xiao et al., 2024). We evaluate the candidates using ASPV signals from each input's first inference under native causal attention, measuring contribution magnitude within each head through attention-weighted projected-value norms (Kobayashi et al., 2020; Zhang et al., 2026).

BOUNDARY assesses candidate sinks from two complementary perspectives. Building upon attention-sink measurements (Gu et al., 2025), the first score accumulates attention-weighted projected-value magnitudes over all receivers that can attend to the source and averages the result across attention heads. The second score employs the last valid token as the receiver, enabling source positions to be compared through a common receiver that has access to the full input (Fu et al., 2025). For source position k at layer ℓ, the two scores are defined as follows:

$$s^{\text{all}}_{\ell,k} = \frac{1}{H}\sum_{h=1}^{H}\left(\sum_{i=k}^{n} A^{\ell,h}_{i,k}\right)\|\mathbf{z}^{\ell,h}_{k}\|_2, \qquad s^{\text{last}}_{\ell,k} = \frac{1}{H}\sum_{h=1}^{H} A^{\ell,h}_{n,k}\|\mathbf{z}^{\ell,h}_{k}\|_2, \tag{2}$$

Here, H is the number of attention heads. We normalize each score by its sum over the content-token positions, and multiply the normalized scores at each position to obtain $E^{\ell}_{k}(x)$. This product favors positions that receive high scores under both receiver settings.

To obtain an input partition, we aggregate the joint signal $E^{\ell}_{k}(x)$ across layers at each source position k and select peaks from the resulting profile. Placing a boundary immediately after each selected peak yields a partition $\mathcal{B}(x) = (B_1, \ldots, B_R)$ into R contiguous blocks, with each selected sink at the end of its block. During the second forward inference, AS-Attention replaces the native causal mask M_{ij} in Eq. 1 with input-adaptive mask

$$\widehat{M}_{ij}(x) = \begin{cases} M_{ij}, & \ell < \ell_{AS}, \\ -\infty, & \ell \geq \ell_{AS},\ j > i,\ \exists r \in \{1,\ldots,R\} :\ i,j \in B_r, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where ℓ_{AS} is the selected attention switching layer. This mask preserves the native causal computation before ℓ_{AS} and enables bidirectional attention across blocks from ℓ_{AS} onward, while retaining direct causal attention within each block. The within-block constraint applies only to direct attention edges; information can still propagate indirectly through other blocks across layers.

3.4 LAYER: ATTENTION SWITCHING AND READOUT SELECTION

LAYER selects where AS-Attention begins and which subsequent layers provide the retrieval representation. Both selections use the unlabeled calibration set $\mathcal{C}$ and remain fixed for each task.

Attention Switching Layer Selection. We aim to preserve early contextual processing while enabling cross-block attention in later layers. Prior work associates early layers with broad information mixing and middle layers with more limited mixing and attention sinks (Queipo-de Llano et al., 2026). This motivates using changes in ASPV signals across layers to determine where AS-Attention begins. Using the first inference under native causal attention, we compute a value-aware sink rate at each layer. For a given source position j, the score for each head is computed by averaging the magnitudes of its attention-weighted projected values across all n receiver positions. The sink rate is the fraction of heads with scores at least τ_V, the contribution threshold:

$$\text{sink-score}^{(\ell,h)} = \frac{1}{n}\sum_{i=1}^{n}\left\|A^{\ell,h}_{i,j}\mathbf{z}^{\ell,h}_{j}\right\|_2, \quad \text{sink-rate}^{(\ell)} = \frac{1}{H}\sum_{h=1}^{H}\mathbf{1}\left[\text{sink-score}^{(\ell,h)} \geq \tau_V\right]. \tag{4}$$

Here τ_V is set to 0.3 (unless otherwise stated). We select the layer with the largest increase relative to the preceding layer:

$$\ell_{AS} = \arg\max_{\ell \in \{2,\ldots,L\}}\left[\text{sink-rate}^{(\ell)} - \text{sink-rate}^{(\ell-1)}\right]. \tag{5}$$

AS-Attention is applied from ℓ_{AS} onward during the second inference.

Readout Layer Selection. Intermediate representations can outperform final-layer outputs on embedding tasks (Skean et al., 2025). We select readout layers based on representation geometry and similarity across layers. After fixing ℓ_{AS}, we run the second inference on $\mathcal{C}$ with AS-Attention and evaluate raw-value representations from candidate layers $\mathcal{L}_{\text{valid}} \subseteq \{\ell_{AS}+1, \ldots, L\}$. For each layer, the spectral score I_ℓ is the stabilized ratio of the mean to the standard deviation of the eigenvalues of its centered raw-value covariance matrix. A higher score favors a more even spectrum.

We also compute the pairwise redundancy $D_{\ell m}$ from the representations of the same calibration inputs in $\mathcal{C}$ to quantify overlap between layers. We initialize the readout set with the valid layer having the highest score I_ℓ. At each subsequent step, we greedily add the remaining valid layer that maximizes its score after penalizing its maximum redundancy with any already selected layer:

$$\mathcal{S}_0 = \left\{ \arg\max_{\ell \in \mathcal{L}_{\text{valid}}} I_\ell \right\}, \quad \mathcal{S}_{i+1} \leftarrow \mathcal{S}_i \cup \left\{ \arg\max_{\ell \in \mathcal{L}_{\text{valid}} \setminus \mathcal{S}_i} \left[I_\ell - \gamma \max_{m \in \mathcal{S}_i} D_{\ell m} \right] \right\}. \tag{6}$$

Here, γ controls the strength of the redundancy penalty. We repeat $K-1$ times to obtain the final readout set $\mathcal{S} = \mathcal{S}_{K-1}$ containing K layers, with $K = 3$ by default.

4 Experiments

The experiments evaluate ASEmbed in terms of retrieval quality, sensitivity to content reordering, and computational cost across natural-language and code retrieval tasks

Table 1: Retrieval performance across code and natural-language retrieval tasks. "+ Ours" denotes composition with ASEmbed. All values are percentages (%). The best result is highlighted in bold.

Method	CSN-CCR		CT-DL		C2T		T2C		C2C		ArguAna		NFCorpus		SciFact		Avg.	
	MRR	NDCG	MRR	NDCG	MRR	NDCG	MRR	NDCG	MRR	NDCG	MRR	NDCG	MRR	NDCG	MRR	NDCG	MRR	NDCG
Mean	32.54	35.35	17.65	29.46	27.34	17.70	35.47	20.08	9.78	8.62	16.99	24.25	8.63	2.52	21.25	23.77	21.21	20.22
Last	0.77	0.79	17.28	28.67	2.73	0.07	3.07	0.80	0.30	0.00	4.35	5.73	5.53	1.49	3.02	2.89	4.63	5.06
SGPT	34.55	37.48	18.89	30.59	28.27	18.45	34.19	18.55	13.52	11.08	13.98	20.13	6.18	1.72	22.03	23.91	21.45	20.24
Echo	42.62	45.68	18.26	29.99	31.97	20.94	33.99	19.64	12.12	11.08	14.71	20.68	9.46	3.30	9.35	9.36	21.56	20.08
ReverseEOL	35.72	38.57	18.50	30.45	49.74	33.14	48.66	37.03	**51.17**	53.58	**23.00**	**33.09**	12.89	4.78	11.30	11.96	31.37	30.33
KV-Embedding	40.25	43.63	16.41	27.90	49.69	33.42	48.39	38.34	34.45	38.44	22.15	31.98	29.24	13.15	46.02	48.02	35.83	34.36
LBNS	50.54	54.29	17.93	30.11	47.58	29.02	46.82	34.17	27.31	18.22	17.22	24.46	8.09	2.56	22.93	25.73	29.80	27.32
LLM2Vec	32.52	35.34	17.43	29.16	27.42	17.74	35.46	20.08	9.78	8.60	4.25	5.48	6.68	2.08	11.43	11.53	18.12	16.25
VA	12.09	13.09	18.16	30.20	50.14	34.29	8.62	4.27	7.70	7.37	17.89	25.75	34.35	19.57	53.97	57.59	25.37	24.02
Ours	**51.58**	**55.48**	**19.45**	**32.07**	**63.60**	**45.44**	**48.82**	**39.44**	49.65	**57.08**	18.57	26.57	**36.67**	**20.95**	**56.10**	**60.01**	**43.06**	**42.13**
Echo + Ours	48.26	51.63	20.03	31.98	52.30	35.86	38.63	27.37	47.80	63.21	14.14	19.89	30.42	15.94	37.94	40.73	36.19	35.83
ReverseEOL + Ours	35.91	38.74	19.39	32.44	69.21	47.18	49.32	38.23	55.48	56.41	22.47	32.54	19.41	8.90	34.27	37.02	38.18	36.43
KV-Emb. + Ours	49.17	52.82	15.80	26.83	66.09	46.03	49.74	29.94	39.69	41.85	20.59	29.41	29.16	14.55	49.37	51.54	39.95	36.62

4.1 Experimental Setup

Retrieval quality is evaluated on eight tasks from MMTEB (Enevoldsen et al., 2025), CoIR (Li et al., 2025), and CoREB (Xue et al., 2026), while sensitivity to content reordering is evaluated on sets constructed from XCodeEval (Khan et al., 2024) and OpenClassGen (Rahman et al., 2026). Retrieval quality is measured using MRR and nDCG@10, and relative variation across reordered inputs is measured using the coefficient of variation (CV).

The baselines comprise four groups: (1) direct extraction, including mean pooling, last-token, and SGPT-style weighted pooling (Muennighoff, 2022); (2) input-based methods, including Echo (Springer et al., 2025) and ReverseEOL (Lin et al., 2026); (3) output-based methods, represented by VA (Zhang et al., 2026); and (4) in-model methods, including LLM2Vec (BehnamGhader et al., 2024), LBNS (Lin et al., 2025) and KV-Embedding (Tang & Yang, 2026). Following prior work (Qin et al., 2025; Tang & Yang, 2026), all methods within each task use the same frozen backbone without embedding-specific training: Qwen2.5-Coder-Instruct-7B (Hui et al., 2024) for code-related tasks and Qwen3-4B (Yang et al., 2025) for the remaining tasks. Appendix C provides details on datasets and baselines.

4.2 Main Results: Retrieval and Order Robustness

Retrieval quality. Table 1 shows the retrieval performance across eight code and natural-language tasks. Compared with the best baseline for each task and metric, ASEmbed improves MRR and

Table 2: Retrieval robustness under relevance-preserving content reordering. CV_M and CV_N denote the dimensionless coefficients of variation for MRR and nDCG across matched reorderings.

Method	XCodeEval				OpenClassGen			
	MRR↑	nDCG↑	CV_M ↓	CV_N ↓	MRR↑	nDCG↑	CV_M ↓	CV_N ↓
Mean	83.42	65.07	0.065	0.078	71.04	75.01	0.002	0.001
Last	34.25	24.62	1.190	1.220	1.35	1.48	0.044	0.020
SGPT	82.42	64.13	0.282	0.297	68.02	71.51	0.002	0.002
Echo	86.04	66.05	**0.012**	**0.007**	34.47	38.10	0.002	0.002
KV-Emb.	55.93	48.53	0.146	0.188	**85.69**	**87.78**	0.004	0.003
RevEOL	70.76	62.49	0.126	0.135	77.62	80.91	0.004	0.004
LBNS	92.77	74.14	0.120	0.136	84.73	87.22	0.045	0.044
LLM2Vec	83.41	65.11	0.064	0.077	70.97	74.96	**0.001**	**0.001**
VA	15.07	13.44	0.671	0.797	22.38	24.55	0.022	0.020
Ours	**95.65**	**86.44**	0.046	0.057	81.24	83.47	0.002	0.002
Echo + Ours	90.93	70.70	0.016	0.025	55.43	57.67	0.005	0.003
KV-Emb. + Ours	61.95	54.55	0.145	0.170	87.63	89.37	0.003	0.002
RevEOL + Ours	94.48	79.80	0.007	0.006	79.94	82.50	0.005	0.003

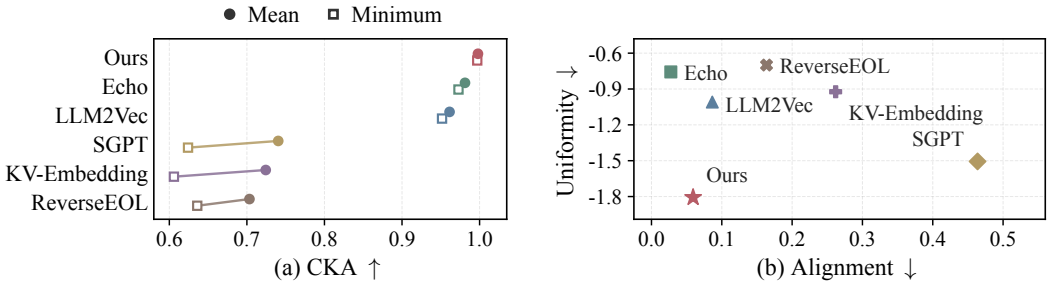

Figure 3: Representation consistency and geometry under content reordering on XCodeEval. (a) Centered kernel alignment (CKA), where higher values indicate greater structural similarity. (b) Alignment and uniformity. Alignment measures the distance between original and reordered query representations, while uniformity characterizes the overall embedding distribution.

NDCG@10 by an average of 1.72 and 1.96 points, respectively. Among baselines, ASEmbed achieves the highest MRR and nDCG@10 on six of the eight tasks. On ArguAna, however, ASEmbed scores below the best baseline on both metrics. One possible explanation is that its contextual aggregation emphasizes shared topics more than the specific stance and rebuttal relations required by this task. To assess compatibility with existing extraction methods, ASEmbed is combined with Echo, ReverseEOL, and KV-Embedding on the same tasks. Each combination improves average MRR and NDCG@10 over its corresponding baseline, with gains of 8.52 and 8.04 points, respectively, averaged equally across the three methods and eight tasks. Echo benefits the most, gaining 14.63 MRR points and 15.74 NDCG@10 points on average. On ArguAna, however, all combinations reduce both metrics, suggesting that combining ASEmbed with these methods does not resolve the difficulty discussed above. **Across the eight tasks, ASEmbed improves average retrieval performance both on its own and in combination with the three extraction methods.**

Order robustness. For each query, the mean retrieval score and CV are computed across its reordered variants, with CV defined as the sample standard deviation divided by the mean. Table 2 illustrates the averages of the per-query means and CVs over the original queries. Among methods evaluated without combination, ASEmbed achieves the highest NDCG@10 averaged over the two evaluation sets (84.96), while its average MRR (88.45) is slightly below that of LBNS (88.75). Combining ASEmbed with Echo, KV-Embedding, or ReverseEOL improves both metrics on both evaluation sets, with average gains over the corresponding baselines ranging from 3.98 to 13.02 MRR points and from 3.81 to 12.11 NDCG@10 points across the two sets. However, the average CV decreases for KV-Embedding but increases for Echo on both evaluation sets. One possible explanation is that content reordering changes the contextual representations and signals obtained from the first causal inference, which may yield different input partitions and uneven gains from the

Table 3: Component-wise replacement analysis on CodeSearchNet-CCR GO.

Component	Variant	nDCG↑	MRR↑
Default		**46.445**	**40.202**
Backbone	Qwen2.5-Coder-1.5B	35.950	31.395
	Qwen2.5-Coder-3B	42.433	36.829
	CodeGemma-7B	43.364	39.862
	OpenCoder-8B	46.845	43.275
Boundary Construction	AST Function	43.760	38.271
	PINE	46.152	39.959
	Fixed Window ($w = 64$)	44.491	38.826
	Fixed Window ($w = 128$)	43.957	38.434
	Fixed Window ($w = 256$)	43.768	38.255
Readout Layer Selection	Empirical	44.466	38.785
	ID Top-3	43.374	37.741
Attention Switching	Bi-direction Attention	30.083	25.965
	Causal Attention	40.667	35.408
	AS-Attention from Layer 0	28.627	24.687
	AS-Attention from Readout Layer	45.116	39.060

subsequent structured attention. **These results show strong retrieval performance under content reordering.** We next examine representation consistency and embedding geometry.

Representation analysis. Figure 3 examines representation consistency and embedding geometry under content reordering on the XCodeEval-derived evaluation set. In Figure 3(a), ASEmbed achieves the highest mean and minimum CKA values among the compared methods (0.9979 and 0.9970, respectively), indicating highly consistent similarity structure across the evaluated comparisons. Figure 3(b) jointly evaluates alignment between original and reordered query representations and uniformity of the overall embedding distribution. ASEmbed combines the second-lowest alignment score of 0.0592 with the lowest uniformity score of -1.8072. Compared with Echo, whose alignment and uniformity scores are 0.0275 and -0.7571, respectively, ASEmbed maintains close representations of original and reordered queries within a more uniform embedding distribution. It also achieves lower scores on both measures than KV-Embedding, indicating closer paired representations and a more uniform distribution. **Together, these results show that ASEmbed combines high structural consistency under content reordering with close paired representations and a comparatively uniform embedding distribution.**

4.3 Component-wise Analysis

Table 3 reports NDCG@10 and MRR on CodeSearchNet-CCR Go for four aspects of ASEmbed: backbone, boundary construction, readout layer selection, and attention switching layer selection.

Backbone. We evaluate ASEmbed with Qwen2.5-Coder models at three parameter scales, as well as CodeGemma-7B and OpenCoder-8B from different model families. Within the Qwen2.5-Coder family, all retrieval metrics improve as model size increases from 1.5B to 7B, suggesting that ASEmbed benefits from greater backbone capacity. Relative to the 1.5B model, the 7B model improves NDCG@10 and MRR by 10.50 and 8.81 points, respectively. The results with CodeGemma-7B and OpenCoder-8B also demonstrate the applicability of ASEmbed to training-free embedding extraction beyond the Qwen2.5-Coder family.

Boundary construction. We compare boundaries derived from ASPV signals with AST-based function boundaries and fixed-size windows, and additionally evaluate PINE with AST-based partitions. ASEmbed achieves the highest NDCG@10, and MRR than these alternatives. Relative to the best fixed-window configuration, $w = 64$, it improves the metrics by 1.95 and 1.38 points, respectively. The corresponding gains over PINE are 0.29 and 0.24 points. Among fixed-size windows, all metrics decrease as the window size increases from 64 to 128 and 256. Larger windows place

more tokens within the same causal block, which may limit the benefit of bidirectional attention across blocks. **These results support using internal signals to construct input partitions for training-free embedding extraction.**

Readout layer selection. We compare the proposed readout layer selection with empirically specified layers and ID Top-3, which selects layers using intrinsic dimensionality. The proposed selection achieves the highest scores on all metrics, exceeding empirical selection by 1.98 NDCG@10 points and 1.42 MRR points, and ID Top-3 by 3.07 and 2.46 points, respectively. **These results support the effectiveness of the proposed criterion for selecting readout layers.**

Attention switching layer selection. To examine when structured attention should begin, attention configurations are compared on CodeSearchNet-CCR Go while keeping boundary construction fixed. The comparison includes native causal attention, bidirectional attention, and AS-Attention starting at layer 0, at the readout layer, or at the selected switching layer. Starting AS-Attention at the selected switching layer achieves the highest nDCG@10 and MRR within this group, exceeding native causal attention by 5.78 and 4.79 points, respectively. Starting at layer 0 yields the lowest scores, possibly because modifying context access in early layers disrupts attention patterns learned during causal pretraining. Starting at the readout layer performs better than starting at layer 0 but remains below the selected switching layer on both metrics. These results favor delaying AS-Attention until the selected layer in the evaluated configuration.

4.4 Additional Analyses and Efficiency

Parameter sensitivity. To assess sensitivity to parameter settings, the sink threshold τ_V, the number of readout layers K, and the calibration set size $|\mathcal{C}|$ are varied (Appendix C.3.1). Figure 6(a) shows that MRR and nDCG@10 increase from the smallest tested threshold, $\tau_V = 0.1$, and remain nearly unchanged at the higher tested values. Both metrics peak at $K = 3$ and decrease as more readout layers are included (Figure 6(b)). Increasing the calibration set from 8 to 96 samples improves both metrics, with smaller successive gains at larger sizes (Figure 6(c)). These results favor three readout layers in this setting and indicate diminishing gains from increasing the calibration set size.

Computational cost. To quantify encoding cost, per-sample latency is compared across input lengths of 128–2048 tokens (Appendix C.3.2). Figure 7 shows that latency increases with input length for all compared methods. ASEmbed has latency comparable to ReverseEOL and higher than LLM2Vec and KV-Embedding across the tested lengths. At 2048 tokens, its latency is approximately 1.31 times that of LLM2Vec and 1.85 times that of KV-Embedding. The two inferences used for boundary construction and representation extraction add encoding cost. Since efficiency is not the main focus of this paper, we leave this as our future work.

5 Conclusion and Limitations

We propose ASEmbed, an ASPV-guided method for training-free retrieval embedding extraction from frozen decoder-only models. The core idea is to use Attention Sink and projected Value evidence to guide how and when to modify attention interactions. Boundary preserves direct causal attention within input-specific blocks while enabling bidirectional access across blocks. Layer uses depth-wise changes in internal patterns to select the intervention onset. Evaluations across backbones and retrieval benchmarks show improvements in retrieval quality and robustness to query content reordering in multiple settings.

However, ASEmbed relies on the frozen model to provide informative internal signals, whose causal relationship with retrieval gains remains unverified. Future work will use controlled interventions to examine their role and investigate how model capabilities and domain differences affect their reliability.

AI USE STATEMENT

We used generative AI tools to polish the manuscript and assist with searches for recent related literature. The authors read and analyzed all cited papers, verified all code and data used in this study, and reviewed all AI-assisted text. We take responsibility for the final content of this work, including its claims and conclusions.

ETHICS STATEMENT

We use publicly available benchmarks, including CoIR https://huggingface.co/CoIR-Retrieval, CoReB https://huggingface.co/datasets/hq-bench/coreb, and MTEB https://huggingface.co/spaces/mteb/leaderboard, to evaluate retrieval performance and representation quality. We also construct order-variation evaluation sets from xCodeEval https://huggingface.co/datasets/NTU-NLP-sg/xCodeEval/tree/main and OpenClassGen https://huggingface.co/datasets/mrahman2025/OpenClassGen to examine the effects of query content reordering. The authors verify the experimental data and review the reordered samples. Both the original code samples and their reordered variants pass the same compilation checks, and the natural-language variants undergo author review. These checks support data quality and consistent comparisons between original and reordered inputs.

REPRODUCIBILITY STATEMENT

We provide our research code through the anonymous repository to support implementation inspection and experimental reproduction. The data and the source code are available in the repository https://anonymous.4open.science/r/ASEmbed. The repository enables readers to examine the representation extraction procedure alongside the method described in the paper and provides a basis for independent verification and further research.

REFERENCES

Federico Barbero, Álvaro Arroyo, Xiangming Gu, Christos Perivolaropoulos, Michael Bronstein, Petar Veličković, and Razvan Pascanu. Why do LLMs attend to the first token?, 2025. URL https://arxiv.org/abs/2504.02732.

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*, 2024. URL https://openreview.net/forum?id=IW1PR7vEBf.

Emily Cheng, Diego Doimo, Corentin Kervadec, Iuri Macocco, Lei Yu, Alessandro Laio, and Marco Baroni. Emergence of a high-dimensional abstraction phase in language transformers. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=0fD3iIBhlV.

Xueying Ding, Xingyue Huang, Mingxuan Ju, Liam Collins, Yozen Liu, Leman Akoglu, Neil Shah, and Tong Zhao. Hierarchical token prepending: Enhancing information flow in decoder-based LLM embeddings. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19047–19066. Association for Computational Linguistics, July 2026. doi: 10.18653/v1/2026.acl-long.870. URL https://aclanthology.org/2026.acl-long.870/.

Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Diganta Misra, Shreeya Dhakal, Jonathan Rystrøm, Roman Solomatin, Ömer Veysel Çağatan, Akash Kundu, Martin Bernstorff, Shitao Xiao, Akshita Sukhlecha, Bhavish Pahwa, Rafał Poświata, Kranthi Kiran GV, Shawon Ashraf, Daniel Auras, Björn Plüster, Jan Philipp Harries, Loïc Magne, Isabelle Mohr, Dawei Zhu, Hippolyte Gisserot-Boukhlef, Tom Aarsen, Jan Kostkan, Konrad Wojtasik, Taemin Lee, Marek Suppa, Crystina Zhang, Roberta Rocca, Mohammed Hamdy, Andrianos Michail, John Yang, Manuel Faysse, Aleksei Vatolin, Nandan Thakur, Manan Dey, Dipam Vasani, Pranjal A Chitale, Simone

Tedeschi, Nguyen Tai, Artem Snegirev, Mariya Hendriksen, Michael Günther, Mengzhou Xia, Weijia Shi, Xing Han Lù, Jordan Clive, Gayatri K, Maksimova Anna, Silvan Wehrli, Maria Tikhonova, Henil Shalin Panchal, Aleksandr Abramov, Malte Ostendorff, Zheng Liu, Simon Clematide, Lester James Validad Miranda, Alena Fenogenova, Guangyu Song, Ruqiya Bin Safi, Wen-Ding Li, Alessia Borghini, Federico Cassano, Lasse Hansen, Sara Hooker, Chenghao Xiao, Vaibhav Adlakha, Orion Weller, Siva Reddy, and Niklas Muennighoff. MMTEB: Massive multilingual text embedding benchmark. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=zl3pfz4VCV.

Lukas Fesser, Mozes Jacobs, Thomas Fel, Andy Keller, and Sham Kakade. A unifying view of attention sinks: Two algorithms, two solutions, 2026. URL https://arxiv.org/abs/2606.08105.

Yuchen Fu, Zifeng Cheng, Zhiwei Jiang, Zhonghui Wang, Yafeng Yin, Zhengliang Li, and Qing Gu. Token prepending: A training-free approach for eliciting better sentence embeddings from LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3168–3181, 2025. doi: 10.18653/v1/2025.acl-long.159. URL https://aclanthology.org/2025.acl-long.159/.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f1b04face60081b689ba740d39ea8f37-Abstract-Conference.html.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report, 2024. URL https://arxiv.org/abs/2409.12186.

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. Scaling sentence embeddings with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 3182–3196. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.181. URL https://aclanthology.org/2024.findings-emnlp.181/.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Mohammad Abdullah Matin Khan, M Saiful Bari, Xuan Long Do, Weishi Wang, Md Rizwan Parvez, and Shafiq Joty. XCodeEval: An execution-based large scale multilingual multitask benchmark for code understanding, generation, translation and retrieval. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6766–6805, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.367. URL https://aclanthology.org/2024.acl-long.367/.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 7057–7075. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.574. URL https://aclanthology.org/2020.emnlp-main.574/.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. NV-embed: Improved techniques for training LLMs as generalist embedding models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=lgsyLSsDRe.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich K"uttler, Mike Lewis, Wen-tau Yih, Tim Rockt"aschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Xiangyang Li, Kuicai Dong, Yi Quan Lee, Wei Xia, Hao Zhang, Xinyi Dai, Yasheng Wang, and Ruiming Tang. CoIR: A comprehensive benchmark for code information retrieval models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22074–22091. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1072. URL https://aclanthology.org/2025.acl-long.1072/.

Ailiang Lin, Zhuoyun Li, Yusong Wang, Keyu Mao, Kotaro Funakoshi, and Manabu Okumura. ReverseEOL: Improving training-free text embeddings via text reversal in decoder-only LLMs. *arXiv preprint arXiv:2606.05858*, 2026. doi: 10.48550/arXiv.2606.05858. URL https://arxiv.org/abs/2606.05858.

Ziyong Lin, Haoyi Wu, Shu Wang, Kewei Tu, Zilong Zheng, and Zixia Jia. Look both ways and no sink: Converting LLMs into text encoders without training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22839–22853, 2025. doi: 10.18653/v1/2025.acl-long.1113. URL https://aclanthology.org/2025.acl-long.1113/.

Reid McIlroy-Young, Katrina Brown, Conlan Olson, Linjun Zhang, and Cynthia Dwork. Order-independence without fine tuning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 72818–72839. Curran Associates, Inc., 2024. doi: 10.52202/079017-2318. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/85529bc995777a74072ef63c05bedd30-Paper-Conference.pdf.

Niklas Muennighoff. Sgpt: Gpt sentence embeddings for semantic search, 2022. URL https://arxiv.org/abs/2202.08904.

Jin Qin, Zihan Liao, Ziyin Zhang, Hang Yu, Peng Di, and Rui Wang. C2llm technical report: A new frontier in code retrieval via adaptive cross-attention pooling, 2025. URL https://arxiv.org/abs/2512.21332.

Enrique Queipo-de Llano, Alvaro Arroyo, Federico Barbero, Xiaowen Dong, Michael M. Bronstein, Yann LeCun, and Ravid Shwartz-Ziv. Attention sinks and compression valleys in LLMs are two sides of the same coin. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1734b19d9afe7d2c7f1154954eaf0d5a-Abstract-Conference.html.

Musfiqur Rahman, SayedHassan Khatoonabadi, and Emad Shihab. Openclassgen: A large-scale corpus of real-world python classes for llm research, 2026. URL https://arxiv.org/abs/2504.15564.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. Parallel context windows for large language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6383–6402, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.352. URL https://aclanthology.org/2023.acl-long.352/.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code Llama: Open foundation models

for code. *arXiv preprint arXiv:2308.12950*, 2023. doi: 10.48550/arXiv.2308.12950. URL https://arxiv.org/abs/2308.12950.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 55854–55875. PMLR, 2025. URL https://proceedings.mlr.press/v267/skean25a.html.

Jacob Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. Repetition improves language model embeddings. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/e9b0c9480b03f9a6bca24ed8c08ee058-Abstract-Conference.html.

Yixuan Tang and Yi Yang. KV-Embedding: Training-free text embedding via internal KV re-routing. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11773–11794. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.540. URL https://aclanthology.org/2026.acl-long.540/.

Chongyang Tao, Tao Shen, Shen Gao, Junshuo Zhang, Zhen Li, Kai Hua, Wenpeng Hu, Zhengwei Tao, and Shuai Ma. LLMs are also effective embedding models: An in-depth overview, July 2025. URL https://arxiv.org/abs/2412.12591. arXiv:2412.12591.

Raghuveer Thirukovalluru and Bhuwan Dhingra. GenEOL: Harnessing the generative power of LLMs for training-free sentence embeddings. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 2295–2308, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.122. URL https://aclanthology.org/2025.findings-naacl.122/.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11897–11916. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.642. URL https://aclanthology.org/2024.acl-long.642/.

Sophie L. Wang, Phillip Isola, and Brian Cheung. The truth lies somewhere in the middle (of the generated tokens). In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. doi: 10.48550/arXiv.2605.09969. URL https://arxiv.org/abs/2605.09969.

Ziqi Wang, Hanlin Zhang, Xiner Li, Kuan-Hao Huang, Chi Han, Shuiwang Ji, Sham Kakade, Hao Peng, and Heng Ji. Eliminating position bias of language models: A mechanistic approach. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 91212–91239, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/e389b15166cf98966ba058965a8c17e3-Paper-Conference.pdf.

Yanzheng Xiang, Hanqi Yan, Lin Gui, and Yulan He. Addressing order sensitivity of in-context demonstration examples in causal language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6467–6481, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.386. URL https://aclanthology.org/2024.findings-acl.386/.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5e5fd18f863cbe6d8ae392a93fd271c9-Abstract-Conference.html.

Yutao Xie, Jiayi Lin, Hande Dong, Lei Zhang, and Zhonghai Wu. Survey of code search based on deep learning. *ACM Transactions on Software Engineering and Methodology*, 33(2):1–42, 2023. doi: 10.1145/3628161. URL https://doi.org/10.1145/3628161.

Siqiao Xue, Zihan Liao, Jin Qin, Ziyin Zhang, Yixiang Mu, Fan Zhou, and Hang Yu. Beyond retrieval: A multitask benchmark and model for code search. *arXiv preprint arXiv:2605.04615*, 2026. URL https://arxiv.org/abs/2605.04615.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Soyoung Yoon, Dongha Ahn, Youngwon Lee, Minkyu Jung, HyungJoo Jang, and Seung-won Hwang. RoToR: Towards more reliable responses for order-invariant inputs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18739–18760, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.918. URL https://aclanthology.org/2025.acl-long.918/.

Yeqin Zhang, Yunfei Wang, Chen Jiaxuan, Ke Qin, Yizheng Zhao, and Cam-Tu Nguyen. LLM-based embeddings: Attention values encode sentence semantics better than hidden states. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=nAfV9H2Z1u.

Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. Dense text retrieval based on pretrained language models: A survey. *ACM Transactions on Information Systems*, 42(4):1–60, 2024. doi: 10.1145/3637870. URL https://doi.org/10.1145/3637870.

Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. PromptReps: Prompting large language models to generate dense and sparse representations for zero-shot document retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4375–4391. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.250. URL https://aclanthology.org/2024.emnlp-main.250/.

APPENDIX CONTENTS

A Instruction Supplement **16**

A.1 Local Attention Sinks and Projected Values 16

A.2 Another Motivation 16

B Method Implementation Details **17**

B.1 Boundary Construction 17

B.2 Readout Layer Selection 18

C Dataset and Baseline Description **18**

C.1 Datasets 18

C.2 Baselines 20

C.3 supplementary experiments result 21

A Instruction Supplement

A.1 Local Attention Sinks and Projected Values

Figure 1(c) shows local attention sinks under two content reorders and changes in their strength across layers. We examine whether these local sinks remain visible after incorporating projected values. The Broadcast Sink interpretation motivates this comparison: an attended source can transmit value information to other token representations (Fesser et al., 2026).

Figure 4 compares received attention and the corresponding ASPV signal in Qwen3-Embedding-8B for the same query shown in Figure 1(c). Several local sinks remain at corresponding token positions in both panels, including within functions and near the transitions between functions. For example, sinks near token positions 181 and 69 remain visible after incorporating projected values.

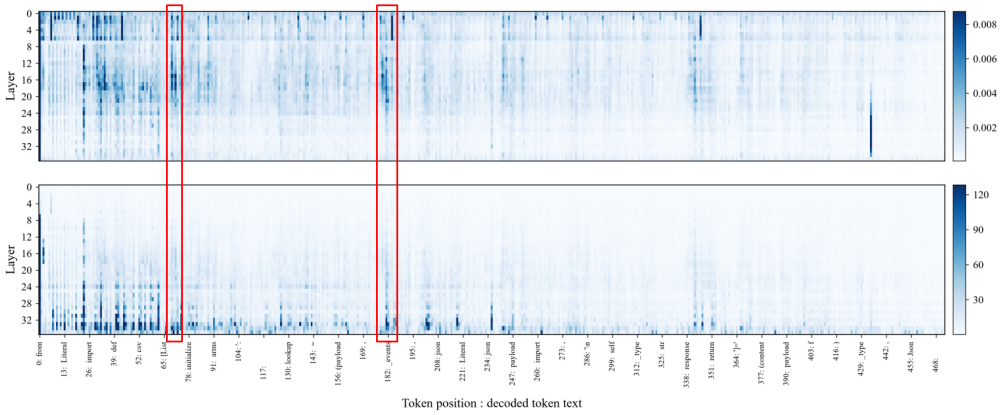

Figure 4: Local attention sinks with and without projected values in Qwen3-Embedding-8B. The upper panel shows received attention $\bar{A}_{\ell,k}$; the lower panel shows the cumulative score $s^{\text{all}}_{\ell,k}$ used by BOUNDARY. Both panels use the same input and zero-based token and layer indices. Each color scale is linear, with limits at the 1st and 99.5th percentiles of its matrix.

A.2 Another Motivation

Connection to learned token aggregation. An independent perspective on the role of attention sinks can be obtained from C2LLM (Qin et al., 2025), which employs Pooling by Multihead Attention (PMA) to construct sequence-level code representations. Instead of uniformly averaging token representations or relying on a single final-token representation, PMA uses a learnable query to aggregate token-level hidden states through cross-attention. The resulting aggregation weights therefore provide an explicit view of which token positions receive greater aggregation mass when constructing the sequence-level embedding.

Interestingly, we observe a clear correspondence between the token positions emphasized by PMA and the attention-sink positions emerging in the frozen decoder-only backbone. As illustrated in Fig. 5, several positions receiving strong PMA aggregation weights also exhibit prominent attention-sink responses across model layers. This correspondence is particularly notable because the two signals arise from different mechanisms: PMA learns how to aggregate token representations for retrieval, whereas attention sinks emerge naturally from the internal attention dynamics of pretrained models (Xiao et al., 2024; Gu et al., 2025).

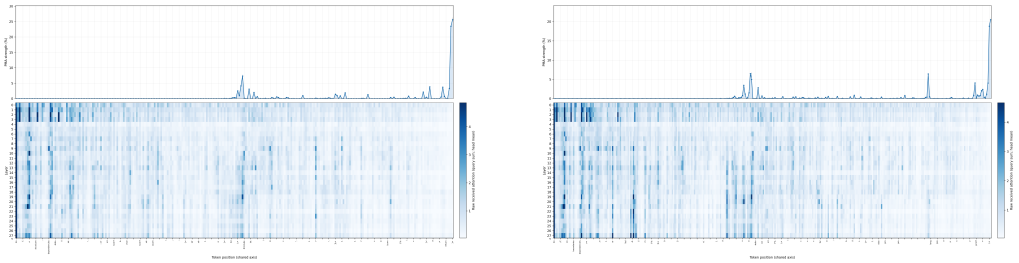

Figure 5: Correspondence between PMA aggregation and attention-sink patterns. Across both examples, token positions receiving relatively large PMA aggregation weights frequently coincide with prominent attention-sink locations across model layers.

Notably, the two mechanisms arrive at similar token preferences from fundamentally different sources: PMA learns where information should be aggregated for sequence-level retrieval representations, whereas attention sinks reveal where the frozen model naturally concentrates its internal information flow. Their empirical correspondence suggests that prominent sink positions are not necessarily arbitrary concentrations of attention, and motivates examining whether they can serve as internal cues for sequence-level representation construction.

This observation provides additional motivation for ASEMBED. Rather than introducing a separately trained token-selection module, ASEMBED exploits these naturally emerging internal signals to identify informative token regions and to organize their interactions during representation formation.

B Method Implementation Details

B.1 Boundary Construction

The main text defines the layer-wise boundary evidence $E_{\ell,j}(x)$ from the two ASPV signals. Here we describe how this continuous evidence is converted into the discrete block partition $B(x)$. The procedure contains three steps.

1. Cross-layer evidence aggregation. The absolute scale of $E_{\ell,j}(x)$ may vary across layers. We therefore normalize each layer-wise profile according to its within-layer ordering and map it to a common reference scale. The normalized profiles are then robustly aggregated across the Boundary layers $\mathcal{L}_B$, yielding a single token-wise boundary profile

$$R_j(x) = \mathrm{Aggregate}_{\ell \in \mathcal{L}_B} E_{\ell,j}(x), \tag{7}$$

where the implementation uses rank-based normalization followed by coordinate-wise median aggregation. The coordinate-wise median emphasizes positions that receive consistently high evidence across layers while suppressing isolated layer-specific responses.

2. Stable boundary localization. We first identify local maxima in $R(x)$ as candidate boundary positions. Weak peaks are removed by adaptive thresholding, and nearby peaks are merged to avoid assigning multiple boundaries to the same response region. We denote the resulting stable peak set by

$$\mathcal{P}(x) = \mathrm{PeakSelect}(R(x)) \,. \tag{8}$$

3. From peaks to blocks. Let the retained peak positions be

$$p_1 < p_2 < \cdots < p_K, \qquad p_k \in \mathcal{P}(x). \tag{9}$$

Each peak is treated as the final token of the current block, so the corresponding cut position is

$$c_k = p_k + 1, \qquad k = 1, \ldots, K. \tag{10}$$

Together with the beginning and end of the valid sequence, these cut positions define the contiguous partition

$$B(x) = (B_1, \ldots, B_{K+1}), \tag{11}$$

which is directly used to construct the AS-Attention visibility matrix in Eq. (3).

B.2 Readout Layer Selection

After the intervention onset is determined as described in the main text, we select a small set of subsequent layers using an unlabeled calibration set $\mathcal{C} = \{x_n\}_{n=1}^{N}$. We score each candidate layer according to its representation spectrum and then discourage selecting layers whose calibration representations are highly redundant with previously selected ones.

Spectral analysis. For each candidate layer ℓ, token-level Raw Value representations are mean-pooled and ℓ_2-normalized for every calibration input,

$$x_n^\ell = \frac{\mathrm{Mean}_{j\in\mathcal{E}_{x_n}} v_j^\ell(x_n)}{\left\|\mathrm{Mean}_{j\in\mathcal{E}_{x_n}} v_j^\ell(x_n)\right\|_2}, \qquad X_\ell = \begin{bmatrix} (x_1^\ell)^\top \\ \vdots \\ (x_N^\ell)^\top \end{bmatrix}. \tag{12}$$

We center X_ℓ over calibration samples and compute its singular values $\{\sigma_{\ell,r}\}$. The corresponding covariance-spectrum values are

$$\lambda_{\ell,r} = \max\left(\frac{\sigma_{\ell,r}^2}{N}, \epsilon\right), \tag{13}$$

from which we define the spectral score

$$I_\ell = \frac{\mathrm{Mean}_r(\lambda_{\ell,r})}{\mathrm{Std}_r(\lambda_{\ell,r}) + \epsilon}. \tag{14}$$

I_ℓ is the inverse coefficient of variation of the covariance spectrum. A larger value indicates that variance is distributed more evenly across spectral directions rather than concentrated in a few dominant components.

Layer selection. Ranking layers solely by I_ℓ may select several neighboring layers with highly similar representations. We quantify the similarity between two candidate layers as

$$D_{\ell m} = \frac{\|X_\ell^\top X_m\|_F}{\|X_\ell\|_F \|X_m\|_F + \epsilon}. \tag{15}$$

The first readout layer is selected by maximizing the spectral score,

$$\ell_1 = \arg\max_{\ell\in\mathcal{L}_{\mathrm{valid}}} I_\ell, \tag{16}$$

and subsequent layers are selected greedily by balancing spectral quality against redundancy with the already selected layers:

$$\ell_t = \arg\max_{\ell\in\mathcal{L}_{\mathrm{valid}}\setminus S_{t-1}} \left[I_\ell - \gamma \max_{m\in S_{t-1}} D_{\ell m}\right], \qquad S_t = S_{t-1}\cup\{\ell_t\}. \tag{17}$$

We use $K = 3$ selected layers in the default configuration.

Readout. The layer-selection criterion is independent of the particular representation readout used after the layers are selected. The selected layers can therefore be combined with different token representations, such as Value or hidden states, and with different fixed cross-layer aggregation operators. In our default implementation, we use the pooled Raw Value representation from each selected layer, normalize each layer independently, and concatenate the K representations:

$$e^\ell(x) = \frac{\mathrm{Mean}_{j\in\mathcal{E}_x} v_j^\ell(x)}{\left\|\mathrm{Mean}_{j\in\mathcal{E}_x} v_j^\ell(x)\right\|_2}, \qquad e(x) = \mathrm{Concat}_{\ell\in S_K}\, e^\ell(x). \tag{18}$$

C Dataset and Baseline Description

C.1 Datasets

We evaluate ASEmbed on established retrieval benchmarks and two evaluation sets constructed for controlled query content reordering. The evaluation covers natural-language retrieval, Text-Code retrieval, Code-Text retrieval, and Code-Code retrieval.

MMTEB. MMTEB (Enevoldsen et al., 2025) is a large-scale multilingual text embedding benchmark containing more than 500 quality-controlled evaluation tasks across over 250 languages. For the natural-language retrieval experiments reported in Table 1, we use the full released data of three English text-to-text retrieval datasets from MMTEB: ArguAna (`https://huggingface.co/datasets/mteb/arguana`), NFCorpus (`https://huggingface.co/datasets/mteb/nfcorpus`), and SciFact (`https://huggingface.co/datasets/mteb/scifact`).

ArguAna evaluates argument retrieval by retrieving the most relevant counterargument for a given argument and contains 1,406 queries and 8,674 candidate documents. NFCorpus is a biomedical information retrieval dataset containing 3,237 queries and 3,633 candidate documents. SciFact evaluates retrieval of scientific evidence relevant to scientific claims and contains 1,109 queries and 5,183 candidate documents. These three datasets constitute the natural-language retrieval tasks reported in Table 1.

CoIR. CoIR (Li et al., 2025) is a comprehensive benchmark for code information retrieval, comprising 10 datasets and eight retrieval subtasks across four main retrieval categories and 14 programming languages. In total, the benchmark contains more than two million code snippets. For the Code-Code retrieval experiments reported in Table 1, we use the full released data of two CoIR datasets: CodeSearchNet-CCR (CSN-CCR; `https://huggingface.co/datasets/CoIR-Retrieval/CodeSearchNet-ccr`) and CodeTransOcean-DL (CT-DL; `https://huggingface.co/datasets/CoIR-Retrieval/codetrans-dl`).

CSN-CCR evaluates code-context retrieval, where an initial segment of a code function is used to retrieve its corresponding remaining segment. It is derived from CodeSearchNet and covers Go, Java, JavaScript, PHP, Python, and Ruby. The released train/dev/test splits contain approximately 905K, 41K, and 53K queries, respectively, with a corpus of approximately one million candidate code segments.

CT-DL evaluates similar-code retrieval across deep-learning frameworks. Given Python code implemented in one framework, the task is to retrieve a semantically equivalent implementation in another framework. Its released train/dev/test splits contain 564, 72, and 180 queries, respectively, with 816 candidate code snippets in total. Both CSN-CCR and CT-DL therefore correspond to the Code-Code retrieval setting reported in Table 1.

CoREB. CoREB (Xue et al., 2026) (`https://huggingface.co/datasets/hq-bench/coreb`) is constructed from rewritten LiveCodeBench programming problems to reduce benchmark contamination. It contains programs in Python, C++, Java, Go, and Ruby and supports T2C, C2T, and C2C retrieval. T2C retrieves correct implementations from problem descriptions, C2T retrieves problem descriptions from implementations, and C2C retrieves semantically equivalent implementations. The evaluation set used here contains 175 programming problems, 1,744 code documents, 875 text documents, and 2,601 retrieval queries.

XCodeEval-derived evaluation set. We construct a controlled C2C retrieval evaluation set from XCodeEval (Khan et al., 2024) (`https://huggingface.co/datasets/NTU-NLP-sg/xCodeEval`) to assess sensitivity to query content reordering. XCodeEval contains approximately 25 million coding examples, totaling 16.5 billion tokens, from about 7,500 programming problems and supports code generation, translation, repair, understanding, and retrieval. Implementations that solve the same problem are treated as semantically related in its C2C retrieval setting.

We use C, C++, Java, and Python examples after deduplication and quality filtering. There are 512 base queries, 3,072 query variants, and 3,584 corpus documents across the four languages. The variants retain the same code content and relevance judgments, and the retrieval corpus remains fixed. Only the order of the snippets changes, allowing us to assess retrieval variation under query content reordering.

OpenClassGen-derived evaluation set. OpenClassGen (Rahman et al., 2026) (`https://huggingface.co/datasets/mrahman2025/OpenClassGen`) contains 324,843 Python classes from 2,970 open-source projects. Each human-written class is paired with a skeleton that re-

tains the class signature, method signatures, and available docstrings while omitting method bodies. The skeleton therefore combines code interfaces with natural-language documentation.

We use these pairs for Code-to-Skeleton retrieval, with the complete class implementation as the query and its corresponding skeleton as the retrieval target. This task evaluates whether the extracted embeddings match a class implementation to a representation of its interfaces and documentation.

We select 694 classes: 354 with three direct methods and 340 with four. All selected classes pass Python AST parsing and compilation checks. The selection also requires relatively balanced method lengths, sufficient method coverage within the class, limited import and header content, and no comments in the query code. Query lengths range from 233 to 2,521 Qwen2.5-Coder tokens, with a median of approximately 541 tokens.

For each class, we generate five fixed variants by reordering its direct methods while preserving the class name, inheritance structure, method contents, and remaining class structure. The resulting evaluation set contains 694 base queries, 3,470 query variants, and 5,008 candidate class skeletons. All variants of a query share the same retrieval target, and the candidate corpus remains fixed. To verify the validity of the reordered samples, each generated variant is parsed using the Python AST and subjected to a compilation check, and variants that fail either check are discarded. Since the reordering operation moves complete method definitions without modifying their contents or the remaining class structure, the resulting variants preserve the same program components and retrieval supervision while differing only in method order.

Input lengths. Across all evaluation datasets, the final tokenized queries and candidate documents, including prompts and any input repetition required by the baselines, do not exceed the maximum input length configured for the corresponding backbone. All inputs are encoded in full, so no input content is lost through truncation.

C.2 Baselines

We compare ASEmbed with methods that extract embeddings from frozen decoder-only LLMs without embedding-specific training. Following the categories in Section 1, we describe input-side, in-model, and output-side methods.

Input-side methods. Echo (Springer et al., 2025) repeats the input and extracts representations from the second occurrence, allowing its tokens to attend to the complete original input under causal attention. ReverseEOL (Lin et al., 2026) encodes the original and reversed inputs and combines their representations to capture complementary context.

In-model methods. These methods modify attention or internal states while keeping the backbone parameters frozen. For LLM2Vec (BehnamGhader et al., 2024), we use only the first conversion step, enabling bidirectional attention without further training. Look Both Ways and No Sink (LBNS) (Lin et al., 2025) combines bidirectional attention with attention-sink suppression. KV-Embedding (Tang & Yang, 2026) reroutes key–value states within the model to provide token positions with sequence-level context.

Output-side methods. We include three direct pooling baselines: mean pooling, last-token pooling, and SGPT-style position-weighted pooling (Muennighoff, 2022). Mean pooling averages token representations uniformly, whereas last-token pooling uses the final token representation, which has access to the full preceding context. SGPT-style pooling assigns larger weights to later token positions. Value Aggregation (VA) (Zhang et al., 2026) aggregates Value representations across token positions and intermediate layers to obtain an embedding.

Combination with ASEmbed. We further combine ASEmbed with representative baselines to examine whether our mechanism is complementary to existing extraction strategies. For **Ours+KV**, we retain the KV-Embedding mechanism and additionally enable our block-wise bidirectional attention. For **Ours+Reverse**, we keep the original reverse-input construction and apply our bidirectional attention during encoding. For **Ours+Echo**, the input is repeated following Echo, while our bidirectional attention is enabled; we further use our adaptive layer selection and readout strategy to obtain the final embedding.

Configuration. Experiments are conducted on either a server with four NVIDIA GeForce RTX 3090 GPUs (24 GB per GPU) or a server with two NVIDIA A100 GPUs (40 GB per GPU). Both servers have 256 GB of system memory. The server configuration is selected according to the computational and memory requirements of the baseline being evaluated.

C.3 SUPPLEMENTARY EXPERIMENTS RESULT

C.3.1 PARAMETER SENSITIVITY

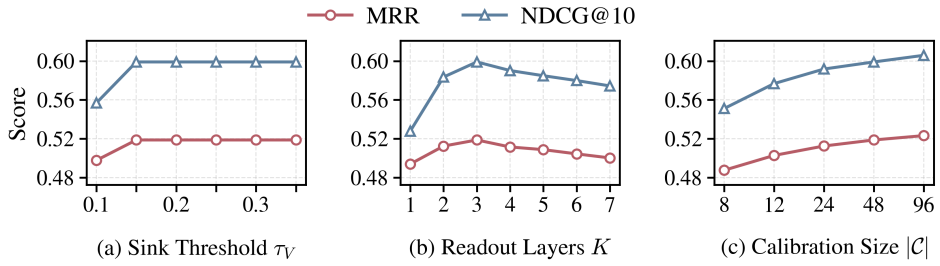

Figure 6: Parameter sensitivity

C.3.2 COMPUTATIONAL COST

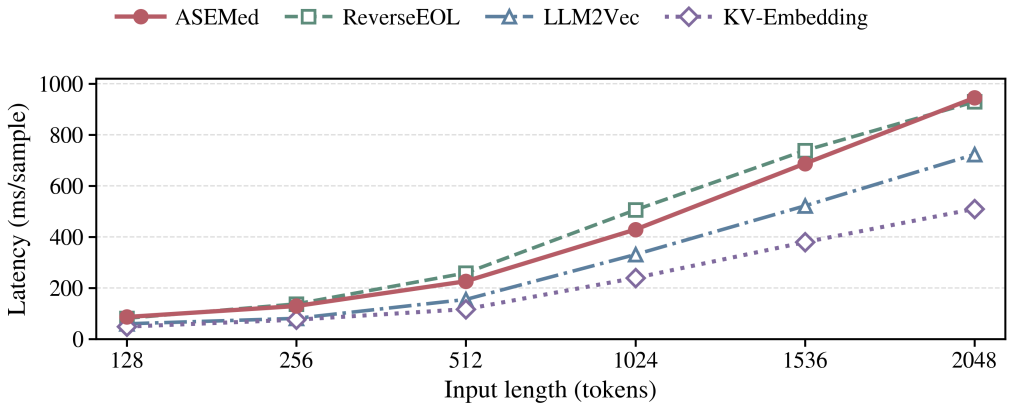

Figure 7: Latency per sample at different input lengths. Lower values indicate faster encoding.