No More K-means: Single-Stage Sparse Coding for Efficient Multi-Vector Retrieval

Lixuan Guo[*1,2] · Yifei Wang[*3] Tiansheng Wen[4,1] Aosong Feng[5] Stefanie Jegelka[6,7] Chenyu You[1]

[1] Stony Brook University [2] Xidian University [3] Amazon AGI SF Lab
[4] Georgia Tech [5] Yale University [6] TUM [7] MIT

ICML 2026 Poster

Lixuan Guo

June 2, 2026

Xi'an, China

The Retrieval Problem

ICML
International Conference
On Machine Learning

Single-Vector Retrieval

- **High efficiency** via simple dot products
- **Struggles** with complex semantic nuances
- Information **compression** loss

The Retrieval Problem

ICML International Conference On Machine Learning

Single-Vector Retrieval

- **High efficiency** via simple dot products
- **Struggles** with complex semantic nuances
- Information **compression** loss

Multi-Vector Retrieval

- Fine-grained **token-level** interactions
- **Superior Retrieval Performance** (e.g., ColBERT)
- Prohibitive **storage** & **index computation**

The Retrieval Problem

ICML International Conference On Machine Learning

Single-Vector Retrieval

- **High efficiency** via simple dot products
- **Struggles** with complex semantic nuances
- Information **compression** loss

Multi-Vector Retrieval

- Fine-grained **token-level** interactions
- **Superior Retrieval Performance** (e.g., ColBERT)
- Prohibitive **storage** & **index computation**

Core Challenge:

Billion-scale token vectors create **severe bottlenecks**:

Storage Explosion

Orders of magnitude larger than SVR

The Retrieval Problem

ICML International Conference On Machine Learning

Single-Vector Retrieval

- **High efficiency** via simple dot products
- **Struggles** with complex semantic nuances
- Information **compression** loss

Multi-Vector Retrieval

- Fine-grained **token-level** interactions
- **Superior Retrieval Performance** (e.g., ColBERT)
- Prohibitive **storage** & **index computation**

Core Challenge:

Billion-scale token vectors create **severe bottlenecks**:

Storage Explosion
Orders of magnitude larger than SVR

Indexing Complexity
K-means on billions of tokens

The Retrieval Problem

ICML International Conference On Machine Learning

Single-Vector Retrieval

- **High efficiency** via simple dot products
- **Struggles** with complex semantic nuances
- Information **compression** loss

Multi-Vector Retrieval

- Fine-grained **token-level** interactions
- **Superior Retrieval Performance** (e.g., ColBERT)
- Prohibitive **storage** & **index computation**

Core Challenge:

Billion-scale token vectors create **severe bottlenecks**:

Storage Explosion
Orders of magnitude larger than SVR

Indexing Complexity
K-means on billions of tokens

Information Loss
Compression sacrifices semantic details

Background: Multi-Vector Retrieval

Multi-Vector Retrieval (MVR):

Token-level embeddings with late interaction

Background: Multi-Vector Retrieval

ICML International Conference On Machine Learning

Multi-Vector Retrieval (MVR):

Token-level embeddings with late interaction

How MVR Works

1 **Token Embedding**

Each token in query and document is encoded into a dense vector

Background: Multi-Vector Retrieval

ICML International Conference On Machine Learning

Multi-Vector Retrieval (MVR):

Token-level embeddings with late interaction

How MVR Works

1. **Token Embedding**
   Each token in query and document is encoded into a dense vector

2. **Late Interaction**
   Compute similarity between all query-document token pairs

Background: Multi-Vector Retrieval

ICML International Conference On Machine Learning

Multi-Vector Retrieval (MVR):

Token-level embeddings with late interaction

How MVR Works

1. **Token Embedding**
   Each token in query and document is encoded into a dense vector
2. **Late Interaction**
   Compute similarity between all query-document token pairs
3. **MaxSim Aggregation**
   Take maximum similarity per query token, then sum

Background: Multi-Vector Retrieval

ICML International Conference On Machine Learning

Multi-Vector Retrieval (MVR):

Token-level embeddings with late interaction

How MVR Works

1. **Token Embedding**
   Each token in query and document is encoded into a dense vector
2. **Late Interaction**
   Compute similarity between all query-document token pairs
3. **MaxSim Aggregation**
   Take maximum similarity per query token, then sum

MaxSim Scoring Function: $$S(Q, D) = \sum_{i=1}^{N} \max_{j=1}^{M} (\mathbf{q}_i \cdot \mathbf{d}_j)$$

This captures **fine-grained semantic alignment** that single-vector retrieval typically misses

Background: Multi-Vector Retrieval

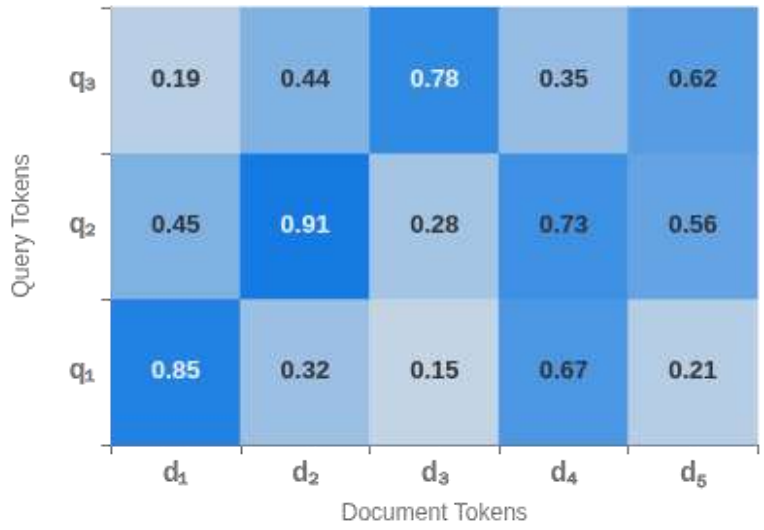
Multi-Vector Retrieval (MVR):

Token-level embeddings with late interaction

How MVR Works

1. **Token Embedding**
   Each token in query and document is encoded into a dense vector
2. **Late Interaction**
   Compute similarity between all query-document token pairs
3. **MaxSim Aggregation**
   Take maximum similarity per query token, then sum

Query-Document Interaction



MaxSim Scoring Function: $S(Q, D) = \sum_{i=1}^{N} \max_{j=1}^{M} (\mathbf{q}_i \cdot \mathbf{d}_j)$

This captures **fine-grained semantic alignment** that single-vector retrieval typically misses

Motivation: The Efficiency Bottleneck

ICML
International Conference
On Machine Learning

Why current MVR systems struggle to scale?

Motivation: The Efficiency Bottleneck

ICML International Conference On Machine Learning

Why current MVR systems struggle to scale?

Storage Cost

Representing documents as **bags of vectors** causes index size to explode by orders of magnitude compared to single-vector baselines.

100x+ larger index than SVR

Motivation: The Efficiency Bottleneck

ICML International Conference On Machine Learning

Why current MVR systems struggle to scale?

Storage Cost

Representing documents as **bags of vectors** causes index size to explode by orders of magnitude compared to single-vector baselines.

100x+ larger index than SVR

Indexing Complexity

State-of-the-art systems rely on **aggressive approximation** strategies like Vector Quantization (VQ) and **large-scale clustering** (K-means).

100+ hrs for billion-scale clustering

Motivation: The Efficiency Bottleneck

ICML International Conference On Machine Learning

Why current MVR systems struggle to scale?

Storage Cost

Representing documents as **bags of vectors** causes index size to explode by orders of magnitude compared to single-vector baselines.

100x+ larger index than SVR

Indexing Complexity

State-of-the-art systems rely on **aggressive approximation** strategies like Vector Quantization (VQ) and **large-scale clustering** (K-means).

100+ hrs for billion-scale clustering

Core Limitation:

the indexing process remains prohibitively complex despite engineering optimization

Existing Approaches in MVR: Three-Stage Pipeline

Standard dense MVR paradigms can be summarized into three stages:

Existing Approaches in MVR: Three-Stage Pipeline

ICML International Conference On Machine Learning

Standard dense MVR paradigms can be summarized into three stages:

I Indexing & Candidate Generation

- → Map document tokens to centroids via **clustering**
- → Build inverted index on centroids
- → Retrieve candidate documents by active centroids

Prunes >98% of corpus

Limitation 1

Clustering is **computationally expensive** and slow

Existing Approaches in MVR: Three-Stage Pipeline

ICML International Conference On Machine Learning

Standard dense MVR paradigms can be summarized into three stages:

I Indexing & Candidate Generation

- → Map document tokens to centroids via **clustering**
- → Build inverted index on centroids
- → Retrieve candidate documents by active centroids

Prunes >98% of corpus

II Approximate Scoring & Pruning

- → Use **compressed representations** (centroids, bit-vectors)
- → Estimate similarity without full decompression
- → Aggressively prune candidate set

Reduces to ~thousands

Limitation 1

Clustering is **computationally expensive** and slow

Limitation 2

Multi-stage pipeline introduces **complex control logic**

Existing Approaches in MVR: Three-Stage Pipeline

ICML International Conference On Machine Learning

Standard dense MVR paradigms can be summarized into three stages:

I Indexing & Candidate Generation

- → Map document tokens to centroids via **clustering**
- → Build inverted index on centroids
- → Retrieve candidate documents by active centroids

Prunes >98% of corpus

II Approximate Scoring & Pruning

- → Use **compressed representations** (centroids, bit-vectors)
- → Estimate similarity without full decompression
- → Aggressively prune candidate set

Reduces to ~thousands

III Final Reranking

- → Reconstruct **full-precision vectors**
- → Compute exact MaxSim scores
- → Resolve minor semantic differences

Final top-k selection

Limitation 1

Clustering is **computationally expensive** and slow

Limitation 2

Multi-stage pipeline introduces **complex control logic**

Limitation 3

MaxSim remains **O(N×M)** quadratic complexity

Our Solution: Single-Stage Sparse Retrieval (SSR)

ICML International Conference On Machine Learning

The Core Idea:

Instead of compressing features into **low-dimensional dense vectors**, we utilize **Sparse Autoencoder (SAE)** to project token embeddings into a **high-dimensional but highly sparse** representation.

Our Solution: Single-Stage Sparse Retrieval (SSR)

ICML International Conference On Machine Learning

The Core Idea:

Instead of compressing features into **low-dimensional dense vectors**, we utilize **Sparse Autoencoder (SAE)** to project token embeddings into a **high-dimensional but highly sparse** representation.

Dense MVR

Low-dim vectors
+ Clustering
+ Multi-stage

SSR (Ours)

High-dim sparse
+ Inverted index
+ Single-stage

Our Solution: Single-Stage Sparse Retrieval (SSR)

ICML International Conference On Machine Learning

The Core Idea:

Instead of compressing features into **low-dimensional dense vectors**, we utilize **Sparse Autoencoder (SAE)** to project token embeddings into a **high-dimensional but highly sparse** representation.

Dense MVR

Low-dim vectors
+ Clustering
+ Multi-stage

SSR (Ours)

High-dim sparse
+ Inverted index
+ Single-stage

Key Advantages

1. **No Clustering**
2. **Natural Inverted Index**
3. **Single-Stage Retrieval**

Our Solution: Single-Stage Sparse Retrieval (SSR)

ICML International Conference On Machine Learning

The Core Idea:

Instead of compressing features into **low-dimensional dense vectors**, we utilize **Sparse Autoencoder (SAE)** to project token embeddings into a **high-dimensional but highly sparse** representation.

Dense MVR

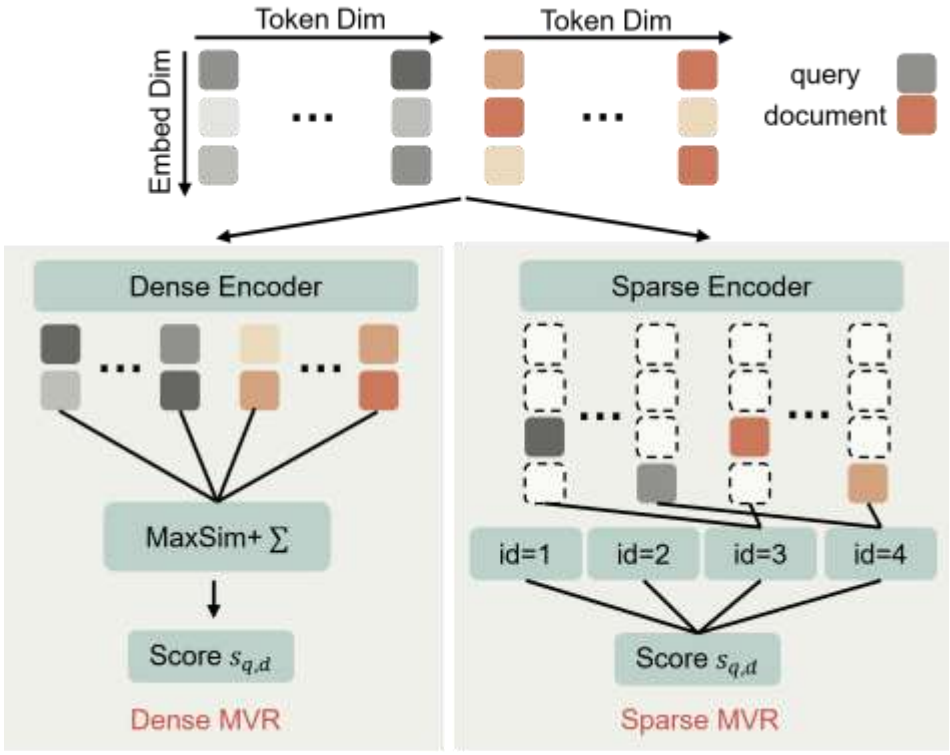
Low-dim vectors
+ Clustering
+ Multi-stage

SSR (Ours)

High-dim sparse
+ Inverted index
+ Single-stage

Key Advantages

1. **No Clustering**
2. **Natural Inverted Index**
3. **Single-Stage Retrieval**



Visualization comparison of two paradigms

Our Solution: Single-Stage Sparse Retrieval (SSR)

ICML International Conference On Machine Learning

The Sparse Scoring Function:

Step 1: SAE Encoding

$$\mathbf{z} = \mathrm{TopK}(\mathbf{W}_{\mathrm{enc}}(\mathbf{x} - \mathbf{b}_{\mathrm{pre}}) + \mathbf{b}_{\mathrm{enc}})$$

Project dense token embedding x into sparse representation z with K active neurons

Step 2: Sparse Interaction

$$S(Q, D) = \sum_{i=1}^{N} \max_{j=1}^{M} \left(\sum_{u \in \mathcal{A}_K(\mathbf{z}_{q_i}) \cap \mathcal{A}_K(\mathbf{z}_{d_j})} \mathbf{z}_{q_i}^{(u)} \cdot \mathbf{z}_{d_j}^{(u)} \right)$$

Interaction only occurs between overlapping activated neurons

Our Solution: Single-Stage Sparse Retrieval (SSR)

ICML International Conference On Machine Learning

The Sparse Scoring Function:

Step 1: SAE Encoding

$$\mathbf{z} = \mathrm{TopK}(\mathbf{W}_{\mathrm{enc}}(\mathbf{x} - \mathbf{b}_{\mathrm{pre}}) + \mathbf{b}_{\mathrm{enc}})$$

Project dense token embedding x into sparse representation z with K active neurons

Step 2: Sparse Interaction

$$S(Q, D) = \sum_{i=1}^{N} \max_{j=1}^{M} \left(\sum_{u \in \mathcal{A}_K(\mathbf{z}_{q_i}) \cap \mathcal{A}_K(\mathbf{z}_{d_j})} \mathbf{z}_{q_i}^{(u)} \cdot \mathbf{z}_{d_j}^{(u)} \right)$$

Interaction only occurs between overlapping activated neurons

Multi-Component Loss

$$\mathcal{L}_{\mathrm{SSR}} = \mathcal{L}_{\mathrm{recon}}(k) + \frac{1}{8}\mathcal{L}_{\mathrm{recon}}(4k) + \alpha\mathcal{L}_{\mathrm{aux}}(k_{\mathrm{aux}}) + \beta\mathcal{L}_{\mathrm{cl}} + \gamma\mathcal{L}_{\mathrm{CE}}$$

Our Solution: Single-Stage Sparse Retrieval (SSR)

ICML International Conference On Machine Learning

The Sparse Scoring Function:

Step 1: SAE Encoding

$$\mathbf{z} = \mathrm{TopK}(\mathbf{W}_{\mathrm{enc}}(\mathbf{x} - \mathbf{b}_{\mathrm{pre}}) + \mathbf{b}_{\mathrm{enc}})$$

Project dense token embedding x into sparse representation z with K active neurons

Step 2: Sparse Interaction

$$S(Q, D) = \sum_{i=1}^{N} \max_{j=1}^{M} \left(\sum_{u \in \mathcal{A}_K(\mathbf{z}_{q_i}) \cap \mathcal{A}_K(\mathbf{z}_{d_j})} \mathbf{z}_{q_i}^{(u)} \cdot \mathbf{z}_{d_j}^{(u)} \right)$$

Interaction only occurs between overlapping activated neurons

Multi-Component Loss

$$\mathcal{L}_{\mathrm{SSR}} = \mathcal{L}_{\mathrm{recon}}(k) + \frac{1}{8}\mathcal{L}_{\mathrm{recon}}(4k) + \alpha\mathcal{L}_{\mathrm{aux}}(k_{\mathrm{aux}}) + \beta\mathcal{L}_{\mathrm{cl}} + \gamma\mathcal{L}_{\mathrm{CE}}$$

1 Reconstruction Loss

Top-K sparse autoencoding with MSE

2 Auxiliary Loss

Reconstruction with inactive neurons

3 Unsupervised CL

Sparse contrastive learning

4 Supervised CL

Cross-entropy with positive-negative pairs

Our Solution: Single-Stage Sparse Retrieval (SSR)

More efficient Retrieval: Neuron-level inverted indexing with coarse-to-fine pruning

Our Solution: Single-Stage Sparse Retrieval (SSR)

ICML International Conference On Machine Learning

More efficient Retrieval: Neuron-level inverted indexing with coarse-to-fine pruning

SSR Base Retrieval

1. **Sparse Projection**
   Encode query tokens into sparse vectors
2. **Identify Active Neurons**
   Extract top-K activated dimensions
3. **Traverse Posting Lists**
   Union of documents from active neurons
4. **Exact Scoring**
   Compute precise MaxSim on candidates

Complexity: O(N · K · L)

N = query tokens, K = sparsity
L = avg posting list length

Our Solution: Single-Stage Sparse Retrieval (SSR)

ICML International Conference On Machine Learning

More efficient Retrieval: Neuron-level inverted indexing with coarse-to-fine pruning

SSR Base Retrieval

1. **Sparse Projection**
   Encode query tokens into sparse vectors
2. **Identify Active Neurons**
   Extract top-K activated dimensions
3. **Traverse Posting Lists**
   Union of documents from active neurons
4. **Exact Scoring**
   Compute precise MaxSim on candidates

Complexity: O(N · K · L)

N = query tokens, K = sparsity
L = avg posting list length

SSR++ Accelerated

Step 1: Coarse Scoring

Extract only principal active neurons (K = 4) for approximate upper-bound scoring

Step 2: Block Skipping

Utilize block upper-bounds to skip blocks that cannot contribute enough

Step 3: Exact Refinement

Compute precise late-interaction score on refined candidate subset

Complexity: O(N · Kcoarse · Lskip)

Kcoarse ≪ K and Lskip ≪ L due to block skipping

Main Results

- *Controlled Setup: Bert-base-uncased as backbone.*

Method	Time	MS	AR	CF	CV	DB	FE	FQ	HQ	NF	NQ	QU	SD	SF	TO	Avg.
Late-interaction dense retrieval																
ColBERT	57.3	36.0	23.6	18.1	67.9	39.2	76.9	31.7	59.2	30.7	51.8	85.7	14.6	66.8	20.5	41.5
COIL	12.6	35.3	29.5	21.7	66.8	39.9	**83.8**	31.3	**71.3**	33.2	52.1	83.8	15.5	70.7	28.1	47.4
ColBERTv2	56.9	39.8	46.3	17.3	73.5	44.9	78.7	36.1	66.6	33.5	56.3	85.3	15.2	69.1	26.1	49.2
PLAID	37.1	39.7	46.1	17.4	73.7	44.7	78.8	36.3	66.7	33.8	56.4	85.0	15.4	69.2	26.3	49.3
AligneR	51.8	38.8	28.9	18.2	68.8	41.6	72.4	33.5	61.5	34.0	52.4	82.3	14.3	70.3	34.8	46.6
CITADEL	15.7	39.9	51.1	18.3	71.5	42.2	76.6	33.0	66.3	33.7	54.0	85.3	15.9	69.0	34.2	49.4
XTR	33.4	45.0	40.7	20.8	73.6	40.8	73.7	34.8	64.7	34.0	52.8	85.9	14.6	71.1	31.3	48.8
Learned sparse retrieval																
Splade-v2	16.3	43.3	47.6	**23.5**	71.5	43.4	78.7	33.8	68.4	33.5	52.1	83.8	15.9	69.4	**36.3**	50.1
Splade-v3	16.6	43.9	**50.8**	23.3	74.8	45.2	79.8	37.5	69.2	35.8	58.6	81.4	15.8	71.0	29.6	51.2
Late-interaction sparse retrieval																
SSR-tok	17.5	45.2	46.1	22.8	76.4	48.2	81.9	39.4	69.5	38.8	60.7	88.3	17.5	72.9	33.1	52.9
SSR-CLS	19.5	**45.5**	46.6	**23.5**	**76.8**	**48.4**	82.8	**40.0**	69.9	**39.1**	**61.2**	**88.7**	**17.9**	**73.3**	33.7	**53.4**

Main Results

ICML International Conference On Machine Learning

- Controlled Setup: Bert-base-uncased as backbone.

Method	Time	MS	AR	CF	CV	DB	FE	FQ	HQ	NF	NQ	QU	SD	SF	TO	Avg.
Late-interaction dense retrieval																
ColBERT	57.3	36.0	23.6	18.1	67.9	39.2	76.9	31.7	59.2	30.7	51.8	85.7	14.6	66.8	20.5	41.5
COIL	12.6	35.3	29.5	21.7	66.8	39.9	**83.8**	31.3	**71.3**	33.2	52.1	83.8	15.5	70.7	28.1	47.4
ColBERTv2	56.9	39.8	46.3	17.3	73.5	44.9	78.7	36.1	66.6	33.5	56.3	85.3	15.2	69.1	26.1	49.2
PLAID	37.1	39.7	46.1	17.4	73.7	44.7	78.8	36.3	66.7	33.8	56.4	85.0	15.4	69.2	26.3	49.3
AligneR	51.8	38.8	28.9	18.2	68.8	41.6	72.4	33.5	61.5	34.0	52.4	82.3	14.3	70.3	34.8	46.6
CITADEL	15.7	39.9	51.1	18.3	71.5	42.2	76.6	33.0	66.3	33.7	54.0	85.3	15.9	69.0	34.2	49.4
XTR	33.4	45.0	40.7	20.8	73.6	40.8	73.7	34.8	64.7	34.0	52.8	85.9	14.6	71.1	31.3	48.8
Learned sparse retrieval																
Splade-v2	16.3	43.3	47.6	**23.5**	71.5	43.4	78.7	33.8	68.4	33.5	52.1	83.8	15.9	69.4	**36.3**	50.1
Splade-v3	16.6	43.9	**50.8**	23.3	74.8	45.2	79.8	37.5	69.2	35.8	58.6	81.4	15.8	71.0	29.6	51.2
Late-interaction sparse retrieval																
SSR-tok	17.5	45.2	46.1	22.8	76.4	48.2	81.9	39.4	69.5	38.8	60.7	88.3	17.5	72.9	33.1	52.9
SSR-CLS	19.5	**45.5**	46.6	**23.5**	**76.8**	**48.4**	82.8	**40.0**	69.9	**39.1**	**61.2**	**88.7**	**17.9**	**73.3**	33.7	**53.4**

Best performance on 9 out of 13 datasets!

Main Results

ICML International Conference On Machine Learning

- Modern Backbone: lama-embed-nemotron-8b

Model	MS	AR	CF	CV	DB	FE	FQ	HQ	NF	NQ	QU	SD	SF	TO	Avg.
llama-8B	61.7	75.7	44.9	89.2	40.1	93.5	61.4	76.7	45.1	66.2	88.1	28.2	82.0	68.8	65.8
Qwen3-8B	**62.4**	**76.9**	46.0	**91.3**	38.8	92.7	64.6	77.0	41.3	64.6	87.8	**29.7**	78.4	**73.0**	66.0
SFR-Embed	59.0	67.2	28.3	87.6	27.3	86.6	60.4	73.8	40.3	34.8	**88.9**	19.9	76.2	55.2	57.5
Linq-Embed	60.5	69.6	30.3	87.1	27.8	87.6	61.2	75.4	41.1	41.9	89.2	21.9	76.4	54.5	58.9
e5-mistral-7b	59.1	61.7	28.5	87.0	34.6	87.0	56.8	73.2	33.4	66.4	88.5	16.3	75.1	55.4	58.8
gte-Qwen2-7B	50.3	54.6	32.2	80.4	**47.3**	90.6	62.0	75.0	40.4	65.6	87.9	23.5	79.5	59.2	60.6
bge-large-v1.5	48.7	64.5	27.3	74.7	41.9	87.3	45.0	75.0	37.1	51.1	**88.9**	22.6	73.6	47.1	56.1
SSR-tok	61.2	75.4	45.5	89.6	41.5	93.9	62.1	77.2	46.0	67.7	87.5	28.1	81.8	69.4	66.4
SSR-CLS	62.0	76.2	**46.4**	90.1	43.3	**94.6**	**62.8**	**77.9**	**46.8**	**68.4**	88.6	29.5	**82.9**	70.5	**67.1**

Main Results

ICML International Conference On Machine Learning

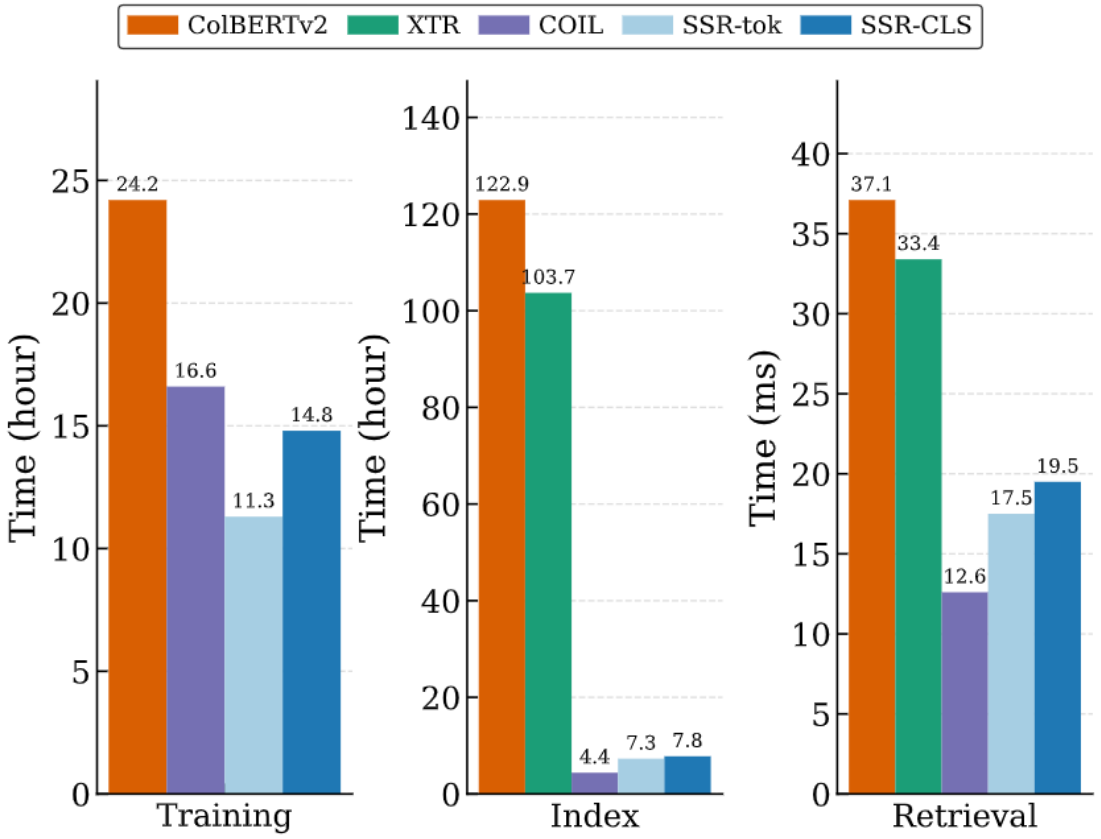
- Modern Backbone: lama-embed-nemotron-8b

Model	MS	AR	CF	CV	DB	FE	FQ	HQ	NF	NQ	QU	SD	SF	TO	Avg.
llama-8B	61.7	75.7	44.9	89.2	40.1	93.5	61.4	76.7	45.1	66.2	88.1	28.2	82.0	68.8	65.8
Qwen3-8B	**62.4**	**76.9**	46.0	**91.3**	38.8	92.7	64.6	77.0	41.3	64.6	87.8	**29.7**	78.4	**73.0**	66.0
SFR-Embed	59.0	67.2	28.3	87.6	27.3	86.6	60.4	73.8	40.3	34.8	**88.9**	19.9	76.2	55.2	57.5
Linq-Embed	60.5	69.6	30.3	87.1	27.8	87.6	61.2	75.4	41.1	41.9	89.2	21.9	76.4	54.5	58.9
e5-mistral-7b	59.1	61.7	28.5	87.0	34.6	87.0	56.8	73.2	33.4	66.4	88.5	16.3	75.1	55.4	58.8
gte-Qwen2-7B	50.3	54.6	32.2	80.4	**47.3**	90.6	62.0	75.0	40.4	65.6	87.9	23.5	79.5	59.2	60.6
bge-large-v1.5	48.7	64.5	27.3	74.7	41.9	87.3	45.0	75.0	37.1	51.1	**88.9**	22.6	73.6	47.1	56.1
SSR-tok	61.2	75.4	45.5	89.6	41.5	93.9	62.1	77.2	46.0	67.7	87.5	28.1	81.8	69.4	66.4
SSR-CLS	62.0	76.2	**46.4**	90.1	43.3	**94.6**	**62.8**	**77.9**	**46.8**	**68.4**	88.6	29.5	**82.9**	70.3	**67.1**

Unlock the rich semantic capacity of modernLLM-based encoder

Main Results

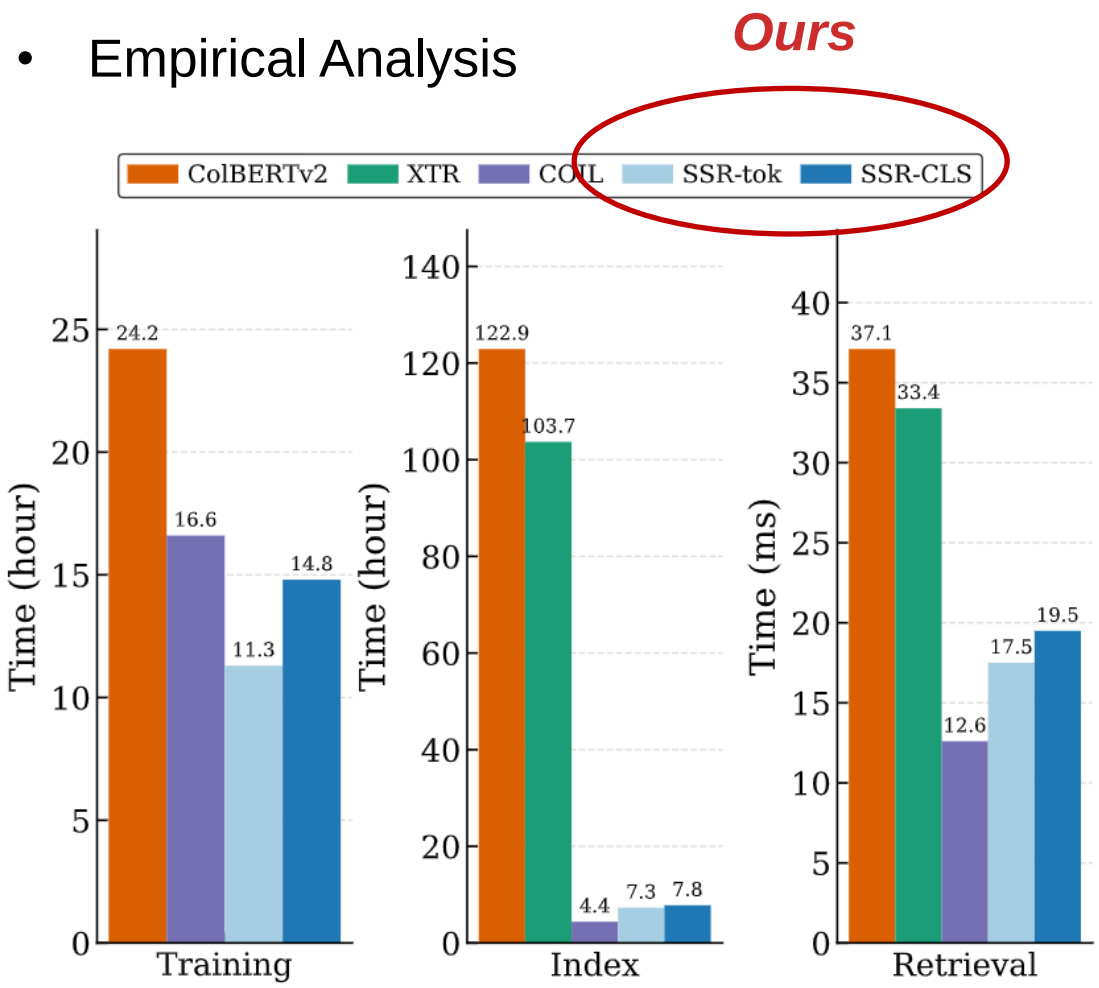
- Empirical Analysis



End-to-End Analysis

Main Results

- Empirical Analysis

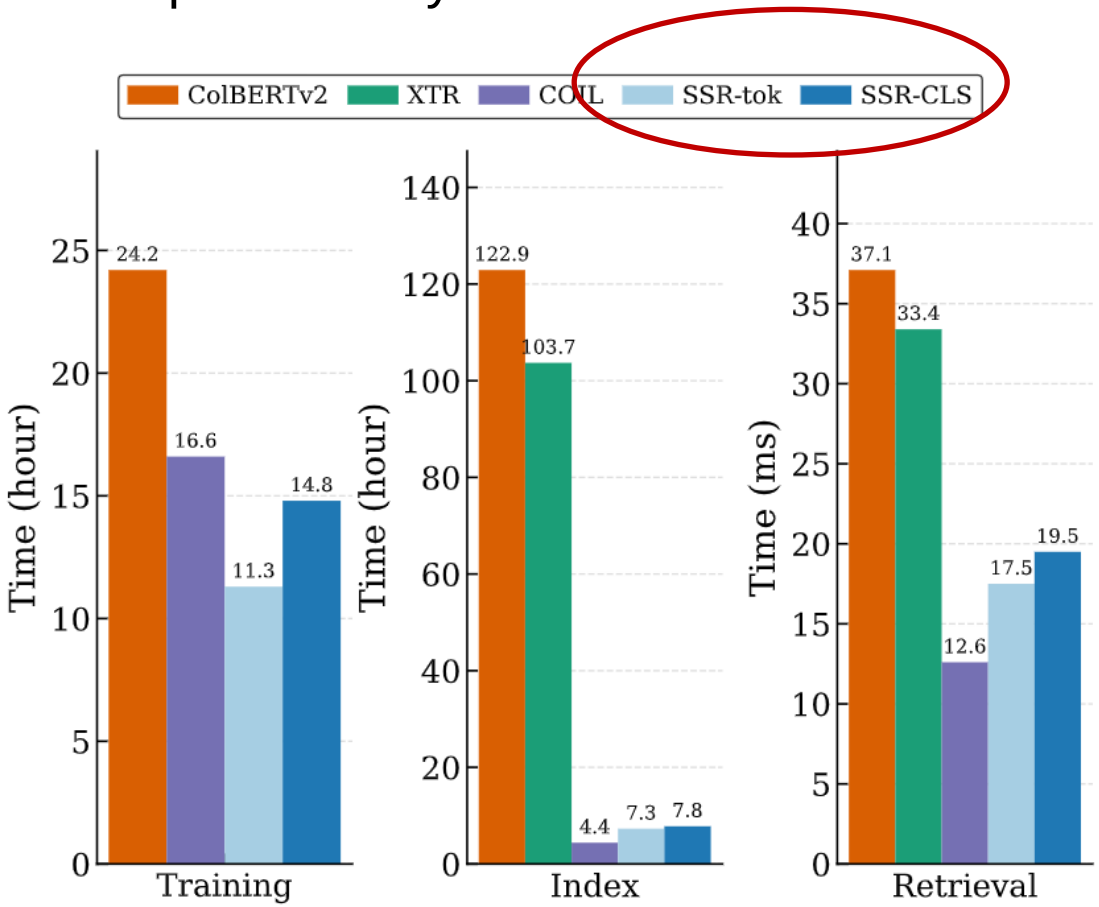


End-to-End Analysis

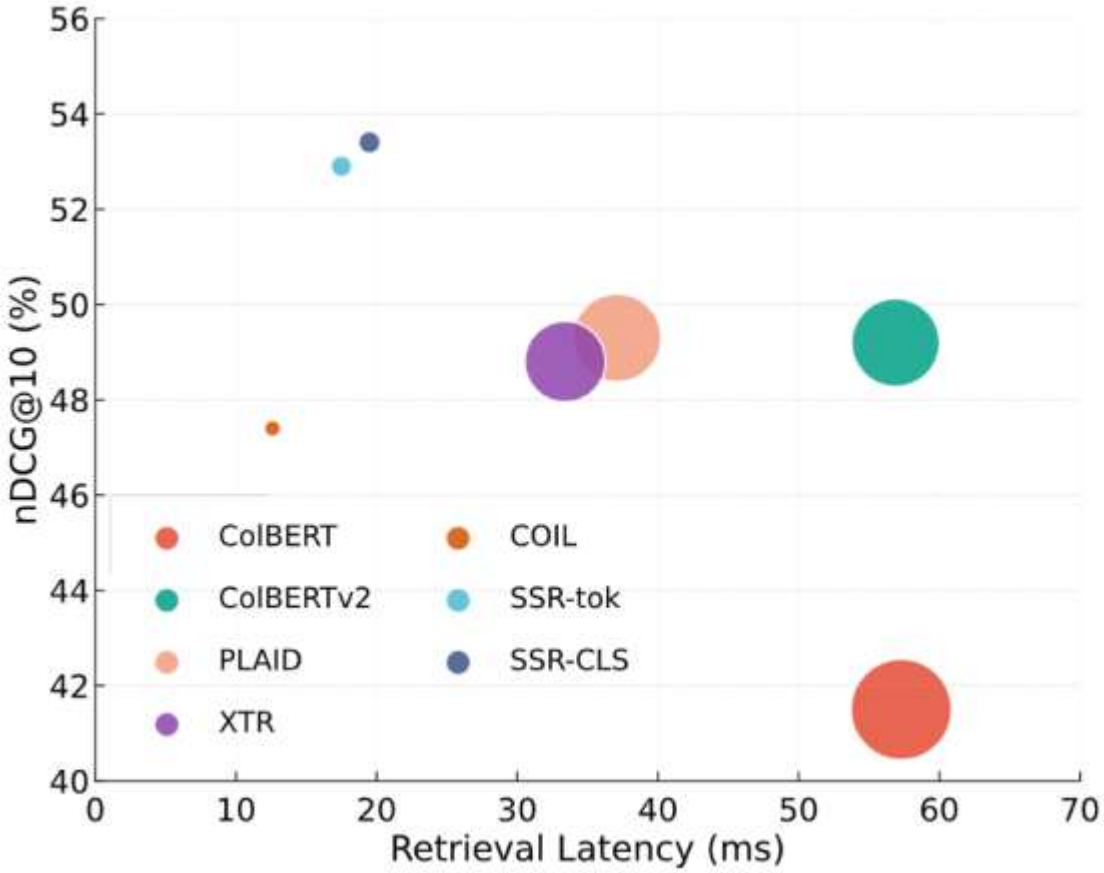
Main Results

- Empirical Analysis

Ours



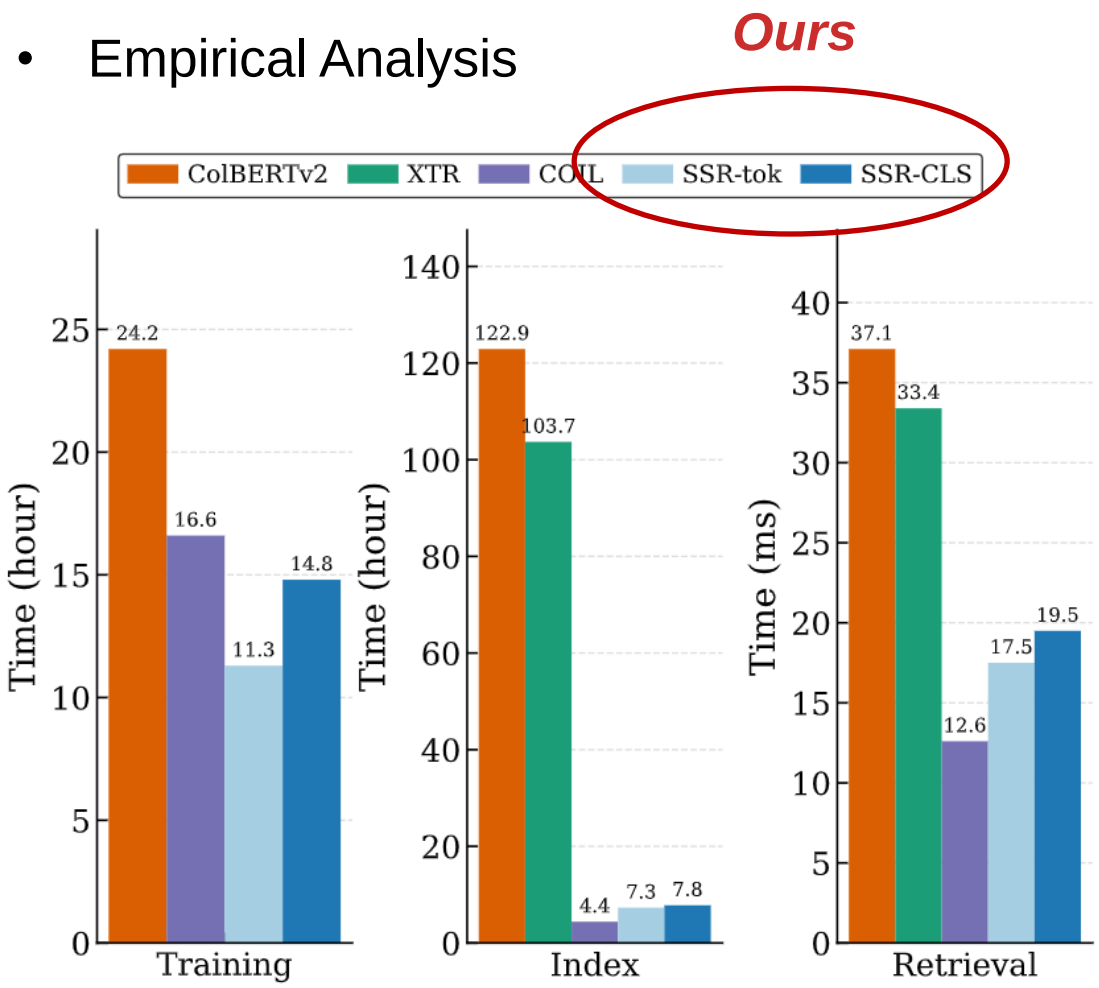
End-to-End Analysis



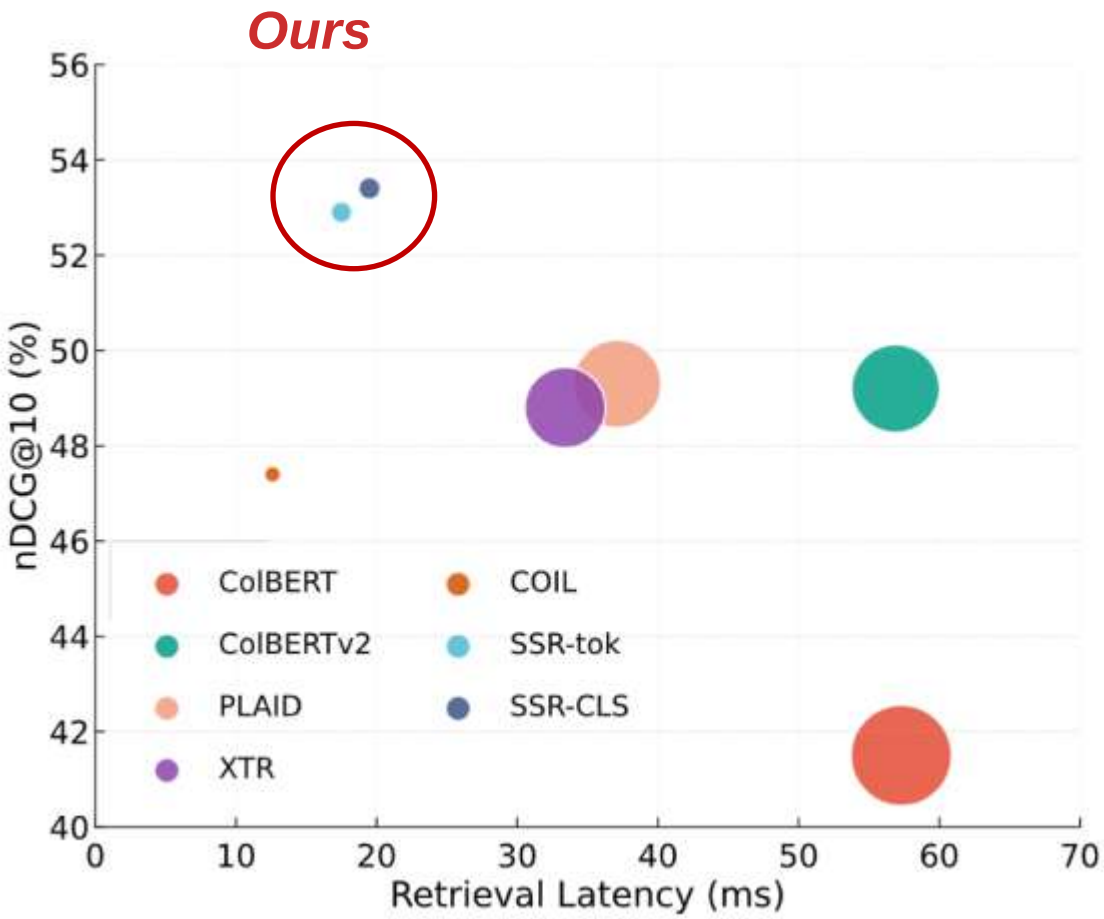
Performance vs. Efficiency

Main Results

- Empirical Analysis



End-to-End Analysis



Performance vs. Efficiency vs. Storage

Thanks for listening!