When Text Embedding Meets Large Language Models: A Comprehensive Survey

Nie et al. (2025)

Presenter: *Haeyoung Lee*

February 25, 2026

Seoul National University

Contents

Notation (1/2)

- **Text space and instances**
  - $\mathcal{X}$: space of text inputs (sentences, queries, passages, documents).
  - $x \in \mathcal{X}$: a single text instance.
  - $q \in \mathcal{X}$: a query; $d \in \mathcal{X}$: a document/passage.
  - $\mathcal{D}$: dataset used for training/evaluation (texts, pairs, or triples).
- **Embedder and embeddings**
  - f_θ: text embedding model with parameters θ.
  - $h = f_\theta(x) \in \mathbb{R}^m$: embedding of x in an m-dimensional space.
  - $h_q = f_\theta(q),\ \ h_d = f_\theta(d)$: query/document embeddings for retrieval.
- **Similarity / retrieval score**
  - $s(\cdot, \cdot)$: similarity in embedding space.
  - Dot product: $s(h_1, h_2) = h_1^\top h_2$
  - Cosine: $s(h_1, h_2) = \dfrac{h_1^\top h_2}{\|h_1\| \, \|h_2\|}$
- **Contrastive samples**
  - Anchor x, positive x^+, negatives $\{x_j^-\}_{j=1}^N$.
  - N: number of negatives per anchor (incl. in-batch negatives).

Notation (2/2)

- **Contrastive samples (Information Retrieval)**
  - Anchor query q, positive document d^+, negatives $\{d_j^-\}_{j=1}^N$.
  - Query–document score: $s(h_q, h_d)$.
- **Decoder-only LLM as embedder**
  - Input token length T; hidden-state dimension m (e.g., 1024/4096).
  - $H \in \mathbb{R}^{T\times m}$: final-layer token hidden states (EOS pooling uses H_T).
  - $P(\cdot)$: pooling operator; $h = P(H) \in \mathbb{R}^m$.
- **LLM-augmented data components**
  - I: instruction / task description; E: in-context examples.
  - Q: query; D^+: positive document; D^-: negative document.
  - L: supervision signal (relevance label, similarity score, or preference/ranking label).
  - *Not all methods use all components (e.g., E often omitted; L may be implicit).*

Motivation

- Text embedding maps a text x to a fixed-length vector $h = f_\theta(x)$ for *fast* semantic comparison.
- Text embeddings are most widely used in **semantic search / information retrieval (IR)**.
- IR is inherently **large-scale**: a query must be scored against a large corpus efficiently.
- Key Challenge: LLMs understand language well, but their *native* representations can be **anisotropic** $\Rightarrow$ similarity may not reflect semantics.

Motivation

- In practice, many IR systems use a **two-stage** pipeline for efficiency:
  1. **Retrieve (fast):** embed query q and documents d, score by $s(h_q, h_d)$, return top-k.
  2. **Rerank / Generate (expensive):** apply a stronger model (cross-encoder or LLM) only to top-k.
- Embedding retrieval is fast because scoring is just **vector similarity**:

$$h_q = f_\theta(q), \quad h_d = f_\theta(d), \quad \text{score} = s(h_q, h_d).$$

- **So the core question is:** how do we train f_θ so that the similarity score matches *true relevance*?

Preliminaries

PLMs $\rightarrow$ Embedding Space Issues

- **PLM era:** "pretraining then fine-tuning" became dominant and boosted many downstream tasks.
- However, vanilla Transformer/PLM embeddings can be *concentrated* in a high-dimensional *conical* space $\Rightarrow$ unexpectedly high similarities for many pairs.
- Result: research focus shifts to **improving embedding spaces** (training objectives, negatives, pooling, post-processing).

Contrastive Learning: Core Objective

Most modern embedders are trained with contrastive objectives (InfoNCE loss):

$$\mathcal{L}_{\mathrm{cl}} = -\mathbb{E}_{x\sim\mathcal{D}} \log \frac{\exp\left(s(h,h^{+})\right)}{\exp\left(s(h,h^{+})\right) + \sum_{j=1}^{N} \exp\left(s(h,h_j^{-})\right)}$$

- $h = f_\theta(x)$, $h^+ = f_\theta(x^+)$, $h_j^- = f_\theta(x_j^-)$.
- Performance is heavily influenced by **negative construction** (hard negatives, mined negatives, false negatives).
- In IR, a common pattern is **two-stage negative mining**:
  - Stage 1: use BM25 top-k as **initial hard negatives** (keyword-overlapping but non-relevant).
  - Stage 2: re-mine with a stronger dense retriever to obtain **harder semantic negatives**.
- **BM25**: a keyword-based (sparse) retriever often used to mine hard negatives.

Non-synthetic Training Datasets

Dataset	Language	Domain	Task	Text-Text / Text-Label Pair
SNLI [30]	English	Web	NLI	550,152
MNLI [31]	English	Web	NLI	392,702
AmazonCounter [46]	Multi	E-commerce	Classification	24,000
Emotion [47]	English	Twitter	Classification	16,000
MTOPIntent [48]	Multi	Web	Classification	18,800
ToxicConversationsClassification [5]	English	Twitter	Classification	50,000
TweetSentimentExtraction [6]	English	Web	Classification	27,500

Dataset	Language	Domain	Task	Queries	Passages	Labels
MS MARCO [29]	English	Web	IR	502,939	8,841,823	532,761
NFCorpus [49]	English	Biomedical	IR	5,922	110,575	3,633
SciFact [50]	English	Scitific	IR	809	920	5,183
BERRI [51]	English	Web	IR	1,013,774	11,187,838	1,013,774
DuReader$_{retrieval}$ [52]	Chinese	Web	IR	97,343	8,096,668	86,395
Multi-CPR-E-commerce [53]	Chinese	E-commerce	IR	100000	1002822	100000
Multi-CPR-Video [53]	Chiense	Video	IR	100000	1000000	100000
Multi-CPR-Biomedical [53]	Chinese	Biomedical	IR	100000	959526	100000
T^2-Ranking [54]	Chinese	Web	IR	258,042	2,303,643	1,613,421
mMARCO [55]	Multi	Web	IR	502,939	8,841,823	532,761
MLDR [56]	Multi	Web	IR	41,434	493,709	41,434
MIRACL [57]	Multi	Web	IR	40,203	90,416,887	343,177
Mr.TyDi [58]	Multi	Web	IR	48,729	58,043,326	49,127
FEVER [59]	English	Web	IR	123,142	5,416,568	140,085
Simclue [7]	Chinese	Web	QA	389,370	2,288,523	775,593
Natural Questions [60]	English	Web	QA	152,148	2,681,468	152,148
SQuAD [61]	English	Web	QA	78,713	23,215	78,713
TriviaQA [62]	English	Web	QA	78,785	78,785	740K
HotpotQA [63]	English	Web	QA	85,000	5,233,329	170,000
FiQA-2018 [64]	English	Finance	QA	5,500	14,166	57,638
BioASQ [65]	English	Biomedical	QA	3,743	35,285	15,559,157
ArchivalQA [66]	English	News	QA	853,644	853,644	483,604
MEDI [67]	English	Web	Multi	1,240,000	1,178,971	1,240,000

LLMs for Embedding

- LLMs bring strong **world knowledge** and **instruction following**, enabling:
    1. **LLM-augmented embedding:** use LLMs to generate data or supervision signals.
    2. **LLM as embedder:** tune an LLM backbone to output high-quality embeddings.
- Most LLM-based embedders use only the final-layer hidden states and discard the decoding head (token-generation head not needed for embedding).
- Decoder-only LLMs are **causal** (left-to-right attention), so the choice of **pooling** matters a lot for sentence-level embeddings.

LLM-Augmented Text Embedding

LLM-Augmented Text Embedding

Two main ways to use LLMs for embedding learning

- **Data synthesis:** directly generate training data (instructions, positives, negatives, labels).
- **Supervision signals:** label/score/filter existing data using LLM judgments.

- Most embedding training data follow the same **contrastive skeleton**:
  - **Information Retrieval (IR):** anchor = query Q, positives/negatives are documents D^+, D^-.
  - **Semantic Textual Similarity (STS):** anchor = text x, positives/negatives are semantically similar/dissimilar texts x^+, x^-.
- With instruction-following embedders, **instruction** I becomes part of the training input.
  - *Instruction lets the model learn a consistent "task mode" even when queries are short or underspecified.*

What can LLMs synthesize?

- **Instructions (I):** generate diverse task descriptions.
- **Positive pairs (Q, D^+ or x, x^+):** create aligned pairs
  - **Doc** $\rightarrow$ **Query**: sample a passage d from an existing corpus, then generate a query q.
- **Negatives (D^- or x^-):** mine/generate **hard negatives** (similar but non-relevant).
- **Labels / similarity scores (L):** LLM-as-judge to **label / score** pairs
  - e.g., relevance yes/no, similarity score.

Key caution

LLMs *increase diversity* and *reduce human labeling cost*, but hard negatives and LLM judging can introduce **false negatives**, so quality control becomes critical.

Examples of LLM-Augmented Methods

LLM generates/labels/filters training examples; **Encoder** is trained on them and used at inference. **Scale** $\approx$ number of constructed examples.

Method	**LLM**	**Encoder**	**Scale**	**Generated data**
DenoSent	ChatGPT	BERT/RoBERTa	1M	(Q, D^+)
SynCSE	ChatGPT/GPT-4	RoBERTa	276K	(Q, D^+, D^-)
Promptagator	FLAN-T5	T5	8M	(Q, D^+)
InPars	ChatGPT	monoT5	10K	(Q, D^+, D^-)
NV-Retriever	(E5-Mistral)	Mistral	956K	(Q, D^+, D^-)
Promptriever	LLaMA3/GPT-4o	LLaMA2	491K	(I, D, D^+, D^-)
E5-Mistral	ChatGPT/GPT-4	Mistral	500K	(I, Q, D^+, D^-)

LLM as Text Embedder

Backbone Selection

- LLM-based embedders reuse open-source LLM backbones.
- Empirical popularity:
  - **Encoder–decoder backbones:** T5 family is dominant.
  - **Decoder-only backbones:** Mistral is most popular, followed by LLaMA and Qwen.
- Typical model size is often **around 7B** parameters (efficiency/performance trade-off).

Input & Pooling Formulation

We view an LLM-based embedder as: **(input construction)** $\rightarrow$ **(hidden states)** $\rightarrow$ **(pooling)**.

- **Base input:** text x (query Q or document/passage D).
- **Optional components:** instruction I, in-context examples E, **prefix tokens** (e.g., ``query:'', ``passage:''), and special token sequence S (e.g., [EOS]).

$$H = f_\theta(I \oplus E \oplus x \oplus S) \in \mathbb{R}^{T \times d} \quad \Rightarrow \quad h = P(H) \in \mathbb{R}^d$$

- T is the token length of the **full input** ($I \oplus E \oplus x \oplus S$).
- Pooling choice (mean/last/EOS) strongly affects sentence/document representations.

Pooling Strategy

- **First pooling:** $h \leftarrow$ first token (e.g., `[CLS]`). Works well with **bi-directional** attention.
- **Mean pooling:** average over token hidden states

$$h = \frac{1}{T} \sum_{t=1}^{T} h_t$$

- **Last / EOS pooling:** $h \leftarrow$ last token (often the `[EOS]` hidden state).
- Decoder-only + **causal** attention: the final token can attend to all previous tokens, so **Last/EOS** (and sometimes Mean) pooling is commonly used.

Optimization

Unsupervised Contrastive Learning (UCL)

- Setting: we have **raw texts** $\{x_i\}$ but **no labeled positive pairs** (no (q, d^+), no (x, x^+)).
- Key idea: create a **positive pair by augmentation** of the same text:

$$x \;\Rightarrow\; (x^{(1)}, x^{(2)}) \quad \text{(two views of the same } x\text{)}$$

- Training: pull the two views together; treat other texts in the batch as negatives (in-batch negatives).
- In the LLM era: prompts/paraphrases can serve as **view generators** (beyond dropout).
- Often used as a **warm-up stage** before stronger supervision (multi-stage training)

Supervised Contrastive Learning (SCL)

- Setting: we have **labeled relationships** such as:
  - Retrieval: (Q, D^+) or (Q, D^+, D^-),
  - Similarity: (x, x^+) (or a similarity label/score).

- Key idea: use labeled positives/negatives to **align the embedding space** with the task:

$$\text{pull } (Q, D^+) \text{ closer,} \quad \text{push } (Q, D^-) \text{ away}$$

Next Token Prediction (NTP) for Embedding

- NTP is the **original pretraining objective** of decoder-only LLMs (predict next token).
- NTP learns token distributions over $\mathbb{R}^{|V|}$, while embeddings live in $\mathbb{R}^{d}$:

$$\text{“dimensionality gap”} \quad \mathbb{R}^{|V|} \not\approx \mathbb{R}^{d}$$

- Why it can still help: to predict next tokens, hidden states must encode rich semantic/context information.
- In embedding training, NTP is typically used **with** contrastive learning:
  - **(A) Multi-task objective:** add NTP as an auxiliary loss

    $$\mathcal{L} = \mathcal{L}_{\mathrm{CL}} + \lambda\, \mathcal{L}_{\mathrm{NTP}}$$

    where $\mathcal{L}_{\mathrm{CL}}$ aligns embedding distances, and $\mathcal{L}_{\mathrm{NTP}}$ helps preserve linguistic information.
  - **(B) Staged training:** NTP pretraining $\rightarrow$ contrastive fine-tuning.

Multi-stage Learning (WCL $\rightarrow$ SCL)

- Many strong embedders adopt a staged recipe:

$$\text{Weakly-supervised CL (WCL)} \rightarrow \text{Supervised CL (SCL)}$$

- **WCL :** massive but noisy supervision signals
  - e.g., web/QA heuristics, neighboring spans, click logs, **LLM-synthetic pairs**
- **SCL :** refine with higher-quality labeled data to match evaluation tasks (IR/STS).

Model Merging (MG)

- **Post-training technique:** combine multiple checkpoints to improve robustness/generalization.

- Why it helps for embeddings:
  - different checkpoints can specialize differently,
  - direct multi-task training can be unstable under data imbalance,
  - merging provides a simple way to find a better balance *without retraining*.

Abbreviations (for next Table)

- **Training paradigms:**
  - **TF**: training-free (no parameter updates)
  - **UCL**: unsupervised contrastive learning
  - **SCL**: supervised contrastive learning
  - **WCL**: weakly-supervised contrastive learning
  - **NTP**: next-token prediction
  - **MS**: multi-stage training (e.g., WCL $\rightarrow$ SCL)
  - **MG**: model merging (combine checkpoints after training)
- **Evaluation settings & tasks:**
  - **ZS**: zero-shot; **FS**: few-shot; **FT**: (full) fine-tuning
  - **IR**: information retrieval
  - **STS**: semantic textual similarity
  - **Uni.**: universal embedding evaluation

Examples of LLM-based Embedders (1/2)

Method	Model	Architecture				Training		Evaluation	
	LLM (Encoder)	Pooling	Attention	Projector	PEFT	Input Format	Paradigm	Setting	Task
Sentence-T5 [168]	T5	Frist / Mean	Bi-Dir	✓	×	-	WCL→SCL	ZS / FT	STS / Clf.
PromptEOL [169]	OPT, LLaMA	Last	Causal	×	×	Prompt	TF / SCL	ZS / ICL	STS / Clf.
MetaEOL [170]	LLaMA, Mistral	Last	Causal	×	×	Prompt	TF	ZS / FT	STS / Clf.
PromptSTH/SUM [171]	OPT, LLaMA, Mistral	Last	Causal	×	×	Prompt	TF	ZS	STS
Token Prepending [172]	LLaMA, Qwen, Gemma	Last	Causal	×	×	Prompt	TF	ZS / FT	STS / Clf.
BeLLM [173]	LLaMA	Last	Bi-Dir	×	LoRA	Prompt	SCL	ZS / FT	STS / Clf.
AutoRegEmbed [174]	LLaMA, Mistral	Special Seq Mean	Causal	×	×	Instruction	SNTP→SCL	ZS	STS
GTR [175]	T5	Mean	Bi-Dir	✓	×	-	WCL→SCL	ZS / FT	IR
SGPT-IR [176]	GPT-Neo	Weighted Mean	Causal	×	Bi-DirtFit	-	SCL	ZS / FT	IR
Llama2Vec [177]	LLaMA	Special Last	Causal	×	LoRA	-	AE+UNTP→SCL	FT	IR
RepLLaMA [178]	LLaMA	Special Last	Causal	×	LoRA	-	SCL	ZS / FT	IR
BMRetriever [179]	Pythia, Gemma, Bi-DiroMistral	Special Last	Causal	×	LoRA	Instruction	WCL→SCL	ZS / FT	IR
ChatRetriever [180]	Qwen	Special Seq Last	Customized	×	LoRA	Instruction	SCL+MLM	ZS / FT	IR
PromptReps [181]	Mistral, Phi-3, LLaMA	Last	Causal	✓	×	Instruction+Prompt	TF / SCL	ZS	IR
LMORT [182]	GPT2, GPT-J	Multi-Layer	Causal	×	×	-	SCL	FT	IR
Mistral-SPLADE [183]	Mistral	Post	Causal	✓	QLoRA	Prompt	SCL+FLOPS	ZS	IR
NV-Retriever [153]	Mistral	Mean	Bi-Dir	×	LoRA	Instruction	SCL→SCL	ZS	IR
Promptriever [155]	LLaMA	Special Last	Causal	×	LoRA	Instruction	SCL	ZS / FT	IR
RARe [184]	LLaMA, Mistral	Mean / Last	Bi-Dir / Causal	×	LoRA	Instruction+Example	SCL	ICL	IR
DEBATER [185]	MiniCPM	Special Seq Last	Causal	×	LoRA	Instruction	SCL+KL	ZS	IR
O1 Embedder [186]	Mistral, LLaMA, Qwen	Gen. Seq Last	Causal	×	LoRA	Instruction	SCL+SNTP	ZS	IR
Search-R3 [187]	Qwen	Gen. Seq Last	Causal	×	LoRA	Instruction	SNTP+KL+SCL+ML→RL	ZS	IR
ReasonEmbed [188]	LLaMA, Qwen	Special Last	Causal	×	LoRA	Instruction	SCL	ZS	IR
Echo [189]	Mistral	Partial Mean	Causal	×	LoRA	Instruction+Prompt	TF / SCL	ZS	Uni.
GenEoL [190]	Mistral	Mean	Causal	×	×	Prompt	TF	ZS	Uni.
MoEE [191]	Deepseek, Qwen, OLMoE	Multi-Layer	Causal	×	×	Prompt	TF	ZS	Uni.
ReBA [192]	GPT-2, LLaMA	Multi-Layer	Causal	×	×	-	TF	ZS	Uni.
LLM2Vec [193]	LLaMA, Mistral	Mean	Bi-Dir	×	LoRA	Instruction	MNTP→UCL/SCL	ZS	Uni.
Cpt [194]	Unknown	Special Last	Unknown	×	×	-	SCL	ZS	Uni.
UDEVER [195]	BLOOM	Special Last	Causal	×	Bi-DirtFit	-	SCL	ZS	Uni.
InstructOR [67]	GTR	Mean	Bi-Dir	✓	×	Instruction	SCL	ZS	Uni.
InBedder [196]	OPT, LLaMA	Last	Causal	×	×	Instruction+Prompt	SNTP	ZS	Uni.
GritLM [197]	Mistral	Mean	Customized	✓ / ×	×	Instruction	SCL+SNTP	ZS / FS	Uni.
MLTP [198]	Mistral	Multi-Layer	Bi-Dir / Causal	×	×	Instruction	SCL	ZS	Uni.
ULLME [199]	LLaMA, Mistral, Phi	Mean	Bi-Dir / Causal	×	LoRA	-	SCL+RL+KL	ZS	Uni.
L^3Prune [200]	LLaMA, Mistral, Qwen, Phi	Weighted Mean	Causal	×	LoRA	Instruction	SCL	ZS	Uni.
LENS [201]	Mistral	Post	Bi-Dir	×	LoRA	Instruction	SCL	ZS	Uni.
DIFFEMBED [202]	Dream	Mean	Bi-Dir	×	LoRA	Instruction	SCL	ZS	Uni.
MGH [203]	Mistral	Weighted Mean	Bi-Dir	×	LoRA	Instruction	SCL	ZS	Uni.
GRACE [204]	LLaMA, Qwen	Hyb. Seq Mean	Causal	×	×	Instruction	RL	ZS	Uni.
Lychee [205]	Qwen	Special Last	Causal	×	LoRA	Instruction	SCL→SCL→MG→SCL	ZS	Uni.
GIRCSE [206]	Mistral, Qwen	Gen. Seq Mean	Causal	×	LoRA	Instruction	SCL+ICR	ZS	Uni.
Anchor [207]	Mistral, LLaMA, Qwen	Special Last	Causal	×	LoRA	Instruction	SNTP→SCL	ZS	Uni.
Text2Token [208]	LLaMA, Mistral	Mean / Last	Bi-Dir / Causal	×	LoRA	- / Prompt	UNTP	ZS	Uni.
Series works by commercial companies									
CoDiEmb (Tecent) [209]	MiniCPM	Mean	Causal	×	×	Instruction	SCL+Pearson+KL+RL→MG	ZS	Uni.
Conan-Embedding-v2 (Tecent) [210]	1.4B LLM	Mean	Causal→Bi-Dir	×	×	Instruction	UNTP→SNTP→WCL→SCL	ZS	Uni.
F2LLM (Ant) [211]	Qwen	Special Last	Causal	×	×	Instruction	SCL	ZS	Uni.
Linq-Embed-Mistral (Linq AI) [212]	Mistral	Special Last	Causal	×	LoRA	Instruction	SCL	ZS	Uni.
QZhou-Embedding (KingSoft) [213]	Qwen	Mean	Bi-Dir	×	×	Instruction	SCL→SCL+ML	ZS	Uni.
Flag Embedding (BAAI)									

Examples of LLM-based Embedders (2/2)

Method	Model	Architecture				Training		Evaluation	
	LLM (Encoder)	Pooling	Attention	Projector	PEFT	Input Format	Paradigm	Setting	Task
BGE-ICL [214]	Mistral, Gemma	Special Last	Bi-Dir / Causal	×	LoRA	Instruction+Example	SCL	ZS / ICL	Uni.
BGE-Multilingual-Gemma2	Gemma	Special Last	Causal	×	×	Instruction	Unknown	ZS	Uni.
E5 Embedding (Microsoft)									
E5-Mistral [157]	Mistral	Special Last	Causal	×	LoRA	Instruction	SCL	ZS	Uni.
SPEED [215]	Mistral	Special Last	Causal	×	LoRA	Instruction	SCL	ZS	
Gemini Embedding (Google)									
Gecko [156]	Unknown	Mean	Unknown	×	×	Instruction	WCL→SCL	ZS	Uni.
Gemini Embedding [216]	Gemini	Mean	Bi-Dir	√	×	Instruction	WCL→SCL→MG	ZS	Uni.
(Nvidia)									
NV-Retriever [153]	Mistral	Mean	Bi-Dir	×	LoRA	Instruction	SCL→SCL	ZS	IR
NV-Embed-v1 [217]	Mistral	Post	Bi-Dir	×	LoRA	Instruction	WCL→SCL	ZS	Uni.
NV-Embed-v2	Mistral	Post	Bi-Dir	×	Unknown	Instruction	WCL→SCL	ZS	Uni.
Qwen Embedding (Alibaba)									
GTE-Qwen2 [218]	Qwen	Special Last	Bi-Dir	×	×	Instruction	WCL→SCL	ZS	Uni.
Qwen3 Embedding [219]	Qwen	Special Last	Causal	×	×	Instruction	WCL→SCL→MG	ZS	Uni.
SFR-Embedding (Salesforce)									
SFR-Mistral [220]	Mistral	Special Last	Causal	×	LoRA	Instruction	SCL	ZS	Uni.
SFR-Embedding-2	Mistral	Special Last	Causal	×	×	Instruction	Unknown	ZS	Uni.

Challenges

False Negatives & Native Embeddings

- **False negative detection :**
  - Real datasets often label only a few positives per query (labeling is expensive) $\Rightarrow$ mined negatives can contain unlabeled positives.
  - False negatives distort contrastive gradients; filtering/sampling/masking strategies are needed.
- **Native high-quality embedding :**
  - Decoder-only LLMs can be anisotropic in their native embedding space (vectors may collapse to similar directions) $\Rightarrow$ "good embeddings for free" are not guaranteed.
  - Open question: can we obtain strong embeddings *without* sacrificing core LLM capabilities?

Privacy Leakage & High-dimensionality

- **Privacy leakage :**
  - Embeddings may reveal sensitive attributes or even enable partial/full text reconstruction (inversion).
  - LLM services + vector DBs increase exposure; security evaluation becomes essential.
- **High-dimensional embeddings :**
  - LLM hidden states often have $d \geq 2048 \Rightarrow$ higher storage and retrieval cost.
  - Direction: efficiency–performance trade-off via **dimension reduction** or **nested representations** (usable at multiple dimensions).

Evaluation Tasks

Evaluation Tasks

- **Semantic Textual Similarity (STS):** predict similarity between two texts
  - gold labels often exist as scores (e.g., 0–5); metric: **Spearman** (rank correlation).
- **Information Retrieval (IR):** retrieve relevant documents for a query
  - key metrics: **Recall@k** (top-k coverage), **MRR** (rank of the first correct result).
- **Universal embedding:** multi-task benchmarks (e.g., **MTEB**) as a "generalization exam".

Key Metrics

- **Spearman (STS): rank correlation**
  Compares the **rank order** of **human-annotated** vs. predicted similarities (-1 to 1; higher is better).

- **Recall@k (IR): top-k coverage**
  @k means "top-k cutoff" (e.g., Recall@10 looks at the top 10 results).

$$\text{Recall@k} = \frac{\#(\text{relevant docs in top-}k)}{\#(\text{all relevant docs})}$$

- **MRR (IR): rank of the first correct result**

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{rank}_i}$$

  where rank_i is the rank position of the **first** relevant doc for query q_i.
  Example: first relevant at rank 2 $\Rightarrow 1/2 = 0.5$ (rank 1 $\Rightarrow$ 1.0).

Part II: Case Study : Qwen3 Embedding & Reranking model

MTEB Leaderboard

- **MTEB** (Massive Text Embedding Benchmark): a broad benchmark for embedding models across retrieval, classification, clustering, reranking, etc.

Rank (Bo…	Model	Zero-shot	Memory Us…	Number of P…	Embedding D…	Max Tokens	Mean (T…	Mean (TaskT…	Bitext …	Classification	Clustering	Instruction R…	Multilabel Class…	Pair Classificat…	Reranking
1	KaLM-Embedding-Gemma3-12B-2511	73%	44884	11.8	3840	32768	**72.32**	**62.51**	**83.76**	77.88	55.77	5.49	33.03	84.73	67.27
2	llama-embed-nemotron-8b	99%	28629	7.5	4096	32768	69.46	61.09	81.72	73.21	54.35	10.82	29.86	83.97	67.78
3	Qwen3-Embedding-8B	99%	14433	7.6	4096	32768	70.58	61.69	80.89	74.00	57.65	10.06	28.66	86.40	65.63
4	gemini-embedding-001	99%			3072	2048	68.37	59.59	79.28	71.82	54.59	5.18	29.16	83.63	65.58
5	Qwen3-Embedding-4B	99%	7671	4.0	2560	32768	69.45	60.86	79.36	72.33	57.15	11.56	26.77	85.05	65.08
6	Octen-Embedding-8B	99%	14433	7.6	4096	32768	67.84	60.28	80.35	66.68	55.68	8.90	25.23	85.12	67.64
7	Seed1.6-embedding-1215	89%			2048	32768	70.26	61.34	78.68	76.75	56.78	-0.02	46.16	85.50	66.24
8	jina-embeddings-v5-text-small	⚠ NA	1137	0.596	1024	32768	67.00	58.90	69.71	71.32	53.41	1.35	41.97	82.93	65.66
9	Qwen3-Embedding-0.6B	99%	1136	0.596	1024	32768	64.34	56.01	72.23	66.83	52.33	5.09	24.59	80.83	61.41
10	jina-embeddings-v5-text-nano	⚠ NA	404	0.212	768	8192	65.52	57.66	67.70	69.18	52.73	0.05	41.31	81.94	64.63

Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models

Qwen3 Embedding Tech Report (2025)

Presenter: *Haeyoung Lee*

February 25, 2026

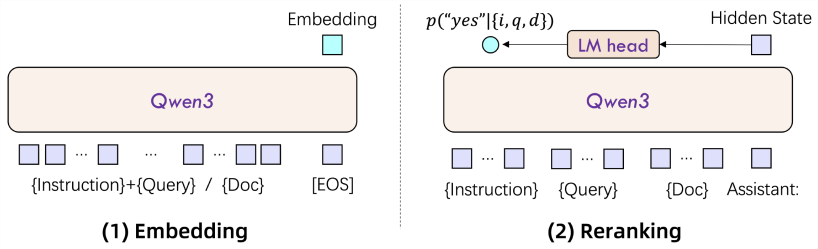
Seoul National University

Overview

- **(1) System:** Embedding (retrieve) vs Reranking (refine)
- **(2) Training recipe:** Stage 1 (**Weakly-supervised Contrastive Learning**) $\rightarrow$ Stage 2 (**Supervised Contrastive Learning / Supervised Fine-Tuning**) $\rightarrow$ Stage 3 (**Model Merging**)
- **(3) Data synthesis:**
  - persona-driven query generation to create realistic multi-intent queries
  - two-step generation: **configuration** $\rightarrow$ **query generation**
  - produce massive (q, d^+) pairs for weak supervision
- **(4) Objectives:** denominator Z_i + masking m_{ij} for false-negative mitigation
- **(5) Results**

Embedding vs. Reranking

- **Embedding (dual-encoder):** encode q and d separately $\Rightarrow$ fast top-k retrieval.
- **Reranking (LLM scoring):** read (I, q, d) jointly $\Rightarrow$ more accurate but slower; applied only to top-k.



Prompt Templates: Embedding vs. Reranking

(1) Embedding input & [EOS] pooling

- Query input (instruction-aware):

$$x_q = I \oplus q \oplus \texttt{<|endoftext|>}$$

- Document input:

$$x_d = d \oplus \texttt{<|endoftext|>}$$

- Decoder-only hidden states:

$$H = f_\theta(x) \in \mathbb{R}^{T \times m}$$

- **[EOS] pooling:** use the final-layer hidden state at <|endoftext|>:

$$h = H_T \in \mathbb{R}^m$$

- Retrieval score: $s(h_q, h_d)$ (cosine).

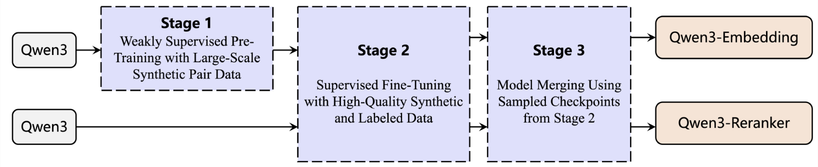
(2) Reranker prompt (Yes/No)

- Joint input (instruction + query + document) using the chat template.
- The model predicts ``yes'' or ``no''; use $p(\text{yes} \mid I, q, d)$ as score.

```
<|im_start|>system
Judge whether the Document meets the requirements based on the Query and the
↪ Instruct provided. Note that the answer can only be "yes" or
↪ "no".<|im_end|>
<|im_start|>user
<Instruct>: {Instruction}
<Query>: {Query}
<Document>: {Document}<|im_end|>
<|im_start|>assistant
<think>\n\n</think>\n\n
```

Multi-stage Training Recipe

- Qwen3 uses a **multi-stage** recipe:
  - **Stage 1 (Weakly-supervised Contrastive Learning):** large-scale synthetic pairs (scale-driven generalization)
  - **Stage 2 (Supervised Contrastive Learning / Supervised Fine-Tuning):** high-quality synthetic + labeled data (task alignment)
  - **Stage 3 (Model Merging):** robustness across tasks/domains



Persona-driven Synthetic Query Generation

- Stage 1 relies on **large-scale synthetic training pairs** generated by a stronger LLM (reported as Qwen3-32B).
- Goal: create **realistic and diverse** queries that match how different users would search for the same passage.
- Core trick: use **persona (character)** to diversify intent and style:
  - the same document can yield different queries depending on who asks and why
  - improves query diversity $\Rightarrow$ better generalization for retrieval/instruction-following
- Output of Stage 1 is mainly **positive pairs** (q, d^+) at massive scale (weak supervision).
- Generator: **Qwen3-32B**; passages sampled from the pretraining corpus.
- Personas: candidates from **PersonaHub** (select top-5 relevant characters).
- Output scale: $\sim$**150M** synthetic weakly-supervised pairs (q, d^+).

Synthetic Data Construction: Overall Flow

1. Sample a passage/document d from a corpus $\Rightarrow$ it will serve as a candidate positive document d^+.
2. Choose a persona/character c (e.g., environmental activist, policymaker, consumer).
3. **Configuration stage:** decide query metadata (type/difficulty/length/language) as JSON.
4. **Query generation stage:** generate the actual query q given (c, d) and the configuration.
5. The final training pair is (q, d^+).

Configuration Stage Template

- Input: **Passage** (candidate document) + **Character** (persona)
- Output: JSON configuration specifying:
  - **Question_Type** (e.g., keywords / summary / yes_or_no / background / acquire_knowledge), **Difficulty** (e.g., high_school / university / phd)

```
Given a **Passage** and **Character**, select the appropriate option from
↪  three fields: Character, Question_Type, Difficulty, and return the output
↪  in JSON format.
First, select the Character who are likely to be interested in the Passage
↪  from the candidates. Then select the Question_Type that the Character
↪  might ask about the Passage; Finally, choose the Difficulty of the
↪  possible question based on the Passage, the Character, and the
↪  Question_Type.
Character: Given by input **Character**

Question_Type:
- keywords: ...
- acquire_knowledge: ...
- summary: ...
- yes_or_no: ...
- background: ...

Difficulty:
- high_school: ...
- university: ...
- phd: ...

Here are some examples
<Example1> <Example2> <Example3>

Now, generate the **output** based on the **Passage** and **Character** from
↪  user, the **Passage** will be in {language} language and the **Character**
↪  will be in English.
Ensure to generate only the JSON output with content in English.

**Passage**:
{passage}
**Character**:
{character}
```

Query Generation Stage Template

- Input: **Character**, **Passage**, and the **Requirement/Configuration** from Stage (1)
- Output: JSON containing the generated query text q
- The query is generated from the persona's perspective and follows the specified type/difficulty/length/language.

```
Given a **Character**, **Passage**, and **Requirement**, generate a query from
↪  the **Character**'s perspective that satisfies the **Requirement** and can
↪  be used to retrieve the **Passage**. Please return the result in JSON
↪  format.

Here is an example:
<example>

Now, generate the **output** based on the **Character**, **Passage** and
↪  **Requirement** from user, the **Passage** will be in {corpus_language}
↪  language, the **Character** and **Requirement** will be in English.
Ensure to generate only the JSON output, with the key in English and the value
↪  in {queries_language} language.

**Character**
{character}
**Passage**
{passage}
**Requirment**
- Type: {type};
- Difficulty: {difficulty};
- Length: the length of the generated sentences should be {length} words;
- Languange: the language in which the results are generated should be
↪  {language} language;
```

From Synthetic Queries to Contrastive Training Data

- After generation, each pair (q, d^+) becomes a **positive pair** for contrastive learning.
- Negatives are then formed by combining:
  - **in-batch negatives:** other positives in the minibatch become negatives for the current query,
  - **explicit hard negatives:** additional $d^-_{i,k}$ (mined or selected) to make training sharper.
- Synthetic data can contain noise/hallucinations $\Rightarrow$ Stage 2 performs high-quality filtering.

Filtering: From Massive to High-quality Synthetic Data

- Stage 1 creates massive synthetic pairs (weakly supervised).
- For Stage 2, they select a **high-quality subset** of synthetic pairs using a similarity-based filter (e.g., keeping pairs above a threshold).
  - Stage 2 also mixes in **human-labeled / curated** data to ground the model and reduce noise.
  - Filter: keep pairs with cosine similarity $> 0.7 \Rightarrow \sim$**12M** high-quality synthetic pairs.
  - Stage 2 also mixes $\sim$**7M** labeled pairs (multi-source).

Notation for Training

- Training instance i (IR-style):

$$(I_i,\ q_i,\ d_i^+,\ \{d_{i,k}^-\}_{k=1}^K)$$

- Query/document embeddings (instruction-aware):

$$h_{q_i} = f_\theta(I_i \oplus q_i), \qquad h_d = f_\theta(d)$$

- Similarity (cosine) with temperature τ:

$$s(a,b) = \frac{a^\top b}{\|a\|\,\|b\|}, \qquad \exp(s(\cdot,\cdot)/\tau)$$

- Z_i: denominator aggregating **positive + hard negatives + in-batch terms**.
- $m_{ik}, m_{ij} \in \{0,1\}$: masks for **false negative mitigation**.

Embedding Objective (1/3): InfoNCE-style Loss

Qwen3-Embedding uses an InfoNCE-style contrastive objective:

$$\mathcal{L}_{\text{emb}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp\big(s(q_i, d_i^+)/\tau\big)}{Z_i}$$

- Numerator: the labeled positive d_i^+ for query q_i.
- Denominator Z_i: **(i) hard negatives** + **(ii) multiple in-batch similarity terms**.

Embedding Objective (2/3): Denominator Z_i

$$
\begin{aligned}
Z_i = \exp\Big(s(q_i, d_i^+)/\tau\Big) + \sum_{k=1}^{K} m_{ik} \exp\Big(s(q_i, d_{i,k}^-)/\tau\Big) + \sum_{j \neq i} m_{ij} \exp(s(q_i, q_j)/\tau) \\
+ \sum_{j \neq i} m_{ij} \exp\Big(s(d_i^+, d_j)/\tau\Big) + \sum_{j \neq i} m_{ij} \exp(s(q_i, d_j)/\tau)
\end{aligned}
$$

- Hard negatives: $\{d_{i,k}^-\}_{k=1}^K$.
- In-batch terms: other queries q_j and documents d_j in the same batch.
- Masks m_{ik}, m_{ij} remove suspicious negatives (next slide).

Embedding Objective (3/3): False Negative Masking

Problem: relevance labels can be incomplete ⇒ "negatives" may contain unlabeled positives.

- If a negative is actually relevant, contrastive learning pushes true positives away.

Mask suspicious negatives:

$$m_{ij} = \begin{cases} 0, & \text{if } s_{ij} > s(q_i, d_i^+) + 0.1 \text{ or } d_j = d_i^+ \\ 1, & \text{otherwise} \end{cases}$$

- s_{ij} denotes the corresponding in-batch score (e.g., $s(q_i, d_j)$ or $s(q_i, q_j)$).

Reranker Objective: Yes/No Scoring

Reranker is trained as supervised classification (SFT) with labels $\ell \in \{\text{yes}, \text{no}\}$:

$$\mathcal{L}_{\text{rerank}} = -\log p(\ell \mid I, q, d)$$

At inference, the rerank score is:

$$\text{score}(q, d) = \frac{\exp(\log p(\text{yes} \mid I, q, d))}{\exp(\log p(\text{yes} \mid I, q, d)) + \exp(\log p(\text{no} \mid I, q, d))}$$

- Cross-encoder style judgement: stronger but applied only to top-k.

Results : MTEB Performance by Task Type

Model	Size	Mean (Task)	Mean (Type)	Bitext Mining	Class-ification	Clus-tering	Inst. Retrieval	Multilabel Class.	Pair Class.	Rerank	Retrieval	STS
Selected Open-Source Models												
NV-Embed-v2	7B	56.29	49.58	57.84	57.29	40.80	1.04	18.63	78.94	63.82	56.72	71.10
GritLM-7B	7B	60.92	53.74	70.53	61.83	49.75	3.45	22.77	79.94	63.78	58.31	73.33
BGE-M3	0.6B	59.56	52.18	79.11	60.35	40.88	-3.11	20.1	80.76	62.79	54.60	74.12
multilingual-e5-large-instruct	0.6B	63.22	55.08	80.13	64.94	50.75	-0.40	22.91	80.86	62.61	57.12	76.81
gte-Qwen2-1.5B-instruct	1.5B	59.45	52.69	62.51	58.32	52.05	0.74	24.02	81.58	62.58	60.78	71.61
gte-Qwen2-7b-Instruct	7B	62.51	55.93	73.92	61.55	52.77	4.94	25.48	85.13	65.55	60.08	73.98
Commercial APIs												
text-embedding-3-large	-	58.93	51.41	62.17	60.27	46.89	-2.68	22.03	79.17	63.89	59.27	71.68
Cohere-embed-multilingual-v3.0	-	61.12	53.23	70.50	62.95	46.89	-1.89	22.74	79.88	64.07	59.16	74.80
Gemini Embedding	-	68.37	59.59	79.28	71.82	54.59	5.18	**29.16**	83.63	65.58	67.71	79.40
Qwen3 Embedding Models												
Qwen3-Embedding-0.6B	0.6B	64.33	56.00	72.22	66.83	52.33	5.09	24.59	80.83	61.41	64.64	76.17
Qwen3-Embedding-4B	4B	69.45	60.86	79.36	72.33	57.15	**11.56**	26.77	85.05	65.08	69.60	80.86
Qwen3-Embedding-8B	8B	**70.58**	**61.69**	**80.89**	**74.00**	**57.65**	10.06	28.66	**86.40**	**65.63**	**70.88**	**81.08**

Results : English / Chinese / Code Benchmarks

Model	Size	Dim	MTEB (Eng, v2)		CMTEB		MTEB (Code)
			Mean (Task)	Mean (Type)	Mean (Task)	Mean (Type)	
Selected Open-Source Models							
NV-Embed-v2	7B	4096	69.81	65.00	63.0	62.0	-
GritLM-7B	7B	4096	67.07	63.22	-	-	73.6^{α}
multilingual-e5-large-instruct	0.6B	1024	65.53	61.21	-	-	65.0^{α}
gte-Qwen2-1.5b-instruct	1.5B	1536	67.20	63.26	67.12	67.79	-
gte-Qwen2-7b-instruct	7B	3584	70.72	65.77	71.62	72.19	56.41^{γ}
Commercial APIs							
text-embedding-3-large	-	3072	66.43	62.15	-	-	58.95^{γ}
cohere-embed-multilingual-v3.0	-	1024	66.01	61.43	-	-	51.94^{γ}
Gemini Embedding	-	3072	73.30	67.67	-	-	74.66^{γ}
Qwen3 Embedding Models							
Qwen3-Embedding-0.6B	0.6B	1024	70.70	64.88	66.33	67.44	75.41
Qwen3-Embedding-4B	4B	2560	74.60	68.09	72.26	73.50	80.06
Qwen3-Embedding-8B	8B	4096	**75.22**	**68.70**	**73.83**	**75.00**	**80.68**

Results : Reranking Models

Model	Param	Basic Relevance Retrieval					
		MTEB-R	CMTEB-R	MMTEB-R	MLDR	MTEB-Code	FollowIR
Qwen3-Embedding-0.6B	0.6B	61.82	71.02	64.64	50.26	75.41	5.09
Jina-multilingual-reranker-v2-base	0.3B	58.22	63.37	63.73	39.66	58.98	-0.68
gte-multilingual-reranker-base	0.3B	59.51	74.08	59.44	66.33	54.18	-1.64
BGE-reranker-v2-m3	0.6B	57.03	72.16	58.36	59.51	41.38	-0.01
Qwen3-Reranker-0.6B	0.6B	65.80	71.31	66.36	67.28	73.42	5.41
Qwen3-Reranker-4B	4B	**69.76**	75.94	72.74	69.97	81.20	**14.84**
Qwen3-Reranker-8B	8B	69.02	**77.45**	**72.94**	**70.19**	**81.22**	8.05

Thank you!