Published: 10 November 2025

CIKM '25: The 34th ACM International Conference on Information and Knowledge Management
November 10 - 14, 2025
Seoul, Republic of Korea

Conference Sponsors:
SIGWEB
SIGIR

SHORT-PAPER

Compressed Concatenation of Small Embedding Models

BEN AYAD MOHAMED AYOUB, University of Passau, Passau, Bayern, Germany
MICHAEL DINZINGER, University of Passau, Passau, Bayern, Germany
KANISHKA GHOSH DASTIDAR, University of Passau, Passau, Bayern, Germany
JELENA MITROVIĆ, University of Passau, Passau, Bayern, Germany
MICHAEL GRANITZER, University of Passau, Passau, Bayern, Germany

Open Access Support provided by:
University of Passau

CIKM '25: Proceedings of the 34th ACM International Conference on Information and Knowledge Management (November 2025)
https://doi.org/10.1145/3746252.3760831
ISBN: 9798400720406

Compressed Concatenation of Small Embedding Models

Ben Ayad Mohamed Ayoub*
University of Passau
Passau, Germany
benaya01@ads.uni-passau.de

Michael Dinzinger
University of Passau
Passau, Germany
michael.dinzinger@uni-passau.de

Kanishka Ghosh Dastidar
University of Passau
Passau, Germany
kanishka.ghoshdastidar@uni-passau.de

Jelena Mitrović
University of Passau
Passau, Germany
jelena.mitrovic@uni-passau.de

Michael Granitzer
University of Passau
Passau, Germany
michael.granitzer@uni-passau.de

Abstract

Embedding models are central to dense retrieval, semantic search, and recommendation systems, but their size often makes them impractical to deploy in resource-constrained environments such as browsers or edge devices. While smaller embedding models offer practical advantages, they typically underperform compared to their larger counterparts. To bridge this gap, we demonstrate that concatenating the raw embedding vectors of multiple small models can outperform a single larger baseline on standard retrieval benchmarks. To overcome the resulting high dimensionality of naive concatenation, we introduce a lightweight unified decoder trained with a Matryoshka Representation Learning (MRL) loss. This decoder maps the high-dimensional joint representation to a low-dimensional space, preserving most of the original performance without fine-tuning the base models. We also show that while concatenating more base models yields diminishing gains, the robustness of the decoder's representation under compression and quantization improves. Our experiments show that, on a subset of MTEB retrieval tasks, our concat-encode-quantize pipeline recovers 89% of the original performance with a 48× compression factor when the pipeline is applied to a concatenation of four small embedding models.

CCS Concepts

• **Information systems → Retrieval models and ranking**.

Keywords

Dense retrieval, Representation Learning, Embedding Models, Quantization

ACM Reference Format:

Ben Ayad Mohamed Ayoub, Michael Dinzinger, Kanishka Ghosh Dastidar, Jelena Mitrović, and Michael Granitzer. 2025. Compressed Concatenation of Small Embedding Models. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3746252.3760831

*Corresponding author

CIKM '25, Seoul, Republic of Korea

ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3760831

1 Introduction

Embedding models have become indispensable to information retrieval, allowing systems to match queries and documents based on semantic content rather than exact keyword overlaps. Recent progress has significantly improved the quality of these embeddings, with noticeable gains in performance on standard benchmarks like BEIR [14] and MTEB [11], but often at the cost of model size and complexity, making them difficult to deploy in edge environments or latency-critical applications.

Although 33 M-parameter models such as E5 [15], GTE [7], and Arctic-Embed [10] (with `snowflake-arctic-embed-s` leading the sub-100 M category as of May 17, 2025) now rival or even surpass larger counterparts on BEIR and MTEB (see the MTEB leaderboard[1]), these smaller models still struggle to capture subtle semantic nuances compared to substantially larger architectures.

Prior work typically tackles this challenge by training increasingly larger single models from scratch —a resource-intensive approach that ignores the complementary strengths of existing models and compounds both computational and environmental costs. We instead repurpose multiple **frozen** embedding models and concatenate their output vectors, producing a unified representation with broader semantic coverage. To address the resulting dimensionality increase, we compress the concatenated representation using a lightweight decoder trained with a Matryoshka Representation Learning (MRL) objective, preserving nearly all of the retrieval performance.

Our contributions can be summarized as follows:

- We demonstrate that concatenating multiple small embedding models can outperform larger single models.
- We propose a lightweight decoder trained with MRL objective, effectively compressing high-dimensional embeddings while preserving performance.
- We show that increasing the number of concatenated models improves the robustness of embeddings at high compression ratios.

[1] https://huggingface.co/spaces/mteb/leaderboard

2 Related work

Open-source embedding models. Recent efforts in open-source embedding models have produced small, efficient models that narrow the performance gap with larger proprietary ones. E5 [15] was the first model pre-trained with a contrastive loss [1] on a large-scale text pair dataset to beat the ranking function BM25 [12] without any fine-tuning. GTE's 110M model variant [7] with multi-stage contrastive learning, outperforms models 10× its size on MTEB. Arctic-Embed [10], with better data sampling, hard negative mining, and improved synthetic query generation, achieved SoTA retrieval performance in many model classes, and highliting the importance of data-centric approaches over pure scale. BGE [17] released a series of models primarily targeting Chinese-language embeddings, widely adopted due to their ease of fine-tuning.

Compression for large-scale retrieval. To enable efficient dense retrieval at scale, a range of embedding compression techniques have been explored. Classical methods, such as product quantization (PQ) [4], and Locality Sensitive Hashing (LSH) [3], decouple encoding from compression, and primarily optimize for reconstruction loss which can limit retrieval performance. Recent work closes that gap by integrating quantization into training, enabling compression while better preserving retrieval accuracy and drastically reducing index size [16, 19]. Knowledge distillation (KD) [2] has also been leveraged to improve embedding compression, with methods explicitly training retrievers from large language models [6], or recent KD-based techniques such as tightly coupled teachers, in-batch interactions, and projective distillation [8, 9, 20].

Adaptive representations. A complementary line of research focuses on adaptive and multi-scale representations. Matryoshka Representation Learning (MRL) [5] learns a single nested embedding such that truncated representations remain informative. Jina-v3 [13] applies MRL to produce highly compressible embeddings down to 32 dimensions, with minimal quality loss. Our work builds on this paradigm by first concatenating the outputs of multiple lightweight embedding models, and then training a decoder with an MRL objective. This yields an adaptively compressible, unified representation that combines the strengths of diverse models while remaining efficient for deployment.

3 Methodology

3.1 Data and architecture:

We follow the same recipe to train all decoders. We use a corpus of $N = 500\,000$ cleaned Wikipedia passages, with each consisting of 512 tokens. Each embedding model S_i of dimensionality d_i defines an embedding matrix of the original Wikipedia passages $E_i \in \mathbb{R}^{N\times d_i}$. Concatenating these feature matrices (w.l.o.g. for S_1 and S_2), results in the following embedding matrix $C = [E_1 \parallel E_2] \in \mathbb{R}^{N\times(d_1+d_2)}$. This formulation yields a standard supervised-learning problem: given the concatenated embeddings $C \in \mathbb{R}^{N\times(d_1+d_2)}$, train a decoder $h : \mathbb{R}^{d_1+d_2} \to \mathbb{R}^{d}, \quad d \ll d_1 + d_2$, to mimic the pairwise similarities of the input (raw concatenation) and output (decoder's output). For the decoder's architecture, we empirically found that a single-layer MLP suffices—any increase in depth led to rapid overfitting.

3.2 Training and Loss Function

We train [2] the decoder to compress concatenated embeddings into a lower-dimensional space while preserving *cosine* pairwise similarities. Let $Z = \{z_j\}_{j=1}^{B} \subset \mathbb{R}^{d_1+d_2}$, be a batch of size B, and let $H = h(Z) \in \mathbb{R}^{B\times d}$, be the decoder's output, with h_i the i-th row of H. We define the similarity loss for a single batch as follows:

$$\ell_{\text{sim}}(H, Z) = \frac{1}{B(B-1)} \sum_{i\neq j} \Big[\cos(h_i, h_j) - \cos(z_i, z_j)\Big]^2,$$

The overall loss that we backpropagate through is defined as the average of the similarity loss (ℓ_{sim}) with varying truncated dimensions of the decoder's outputs. We truncate following a set of Matryoshka dimensions (or stops), $\mathcal{D} = \{d^{(1)}, \dots, d^{(K)}\}$:

$$\mathcal{L} = \frac{1}{|\mathcal{D}|} \sum_{i\in\{1,\dots,K\}} \ell_{\text{sim}}\big(H[:,:\, d^{(i)}],\, Z\big).$$

where $H[:,:\ j]$ denotes the first j columns of H. Each stop explicitly encourage the decoder to enhance its representation at it.

Decoder's output dimension. Unless stated as a subscript (e.g., Dec_{1024}), all decoders produce a 768-dim output, and trained with the following MRL stops: $\{32,\ 64,\ 128,\ 200,\ 256,\ 300,\ 384,\ 512,\ 768\}$.

Overall size. The largest combination M_1S_i totals 142 M parameters (109M for M_1, and 33M S_1). The decoder adds a negligible size $\big(1152 \times 768 \approx 0.9\text{ M}\big)$, which brings the total size to $\approx$ 142.9 M.

3.3 Quantization

Quantization has two stages. **Offline calibration** learns per-dimension break-points on a reference set; **online inference** reuses those break-points to map new decoder outputs to b-bit codes.

Offline calibration. Draw $S = 100\,000$ Wikipedia passages, form their concatenated embeddings $Z_{\text{ref}} \in \mathbb{R}^{S\times(d_1+d_2)}$, and encode them with the best decoder, $H_{\text{ref}} = h(Z_{\text{ref}}) \in \mathbb{R}^{S\times d}$. For each output dimension $j = 1, \dots, d$ compute the $(2^b - 1)$ empirical percentiles

$$\tau_{j,k} \;=\; \text{Percentile}\big(H_{\text{ref},:,j},\, 100\,k/2^b\big), \quad k = 1, \dots, 2^b - 1.$$

Define $\tau_{j,0} = \min H_{\text{ref},:,j}$ and $\tau_{j,2^b} = \max H_{\text{ref},:,j}$. The set $\{\tau_{j,k}\}_{k=0}^{2^b}$ partitions the axis into 2^b equal-mass buckets.

Online inference. Given any new batch of embeddings $Z' \in \mathbb{R}^{N\times(d_1+d_2)}$, we first evaluate the decoder, $H' \;=\; h(Z') \in \mathbb{R}^{N\times d}$, and then assign a b-bit symbol to every coordinate via

$$q_{i,j} \;=\; \sum_{k=1}^{2^b-1} \big[\, H'_{i,j} > \tau_{j,k} \big], \qquad i = 1, \dots, N,\ j = 1, \dots, d.$$

3.4 Retrieval Evaluation

We evaluate our embedding models on downstream retrieval using the Massive Text Embedding Benchmark (MTEB) [11], focusing on a subset of the BEIR [14]. We assess our models on six heterogeneous tasks: NFCorpus (clinical notes to consumer FAQs), SciFact (claim verification in scientific abstracts), ArguAna (argument retrieval

[2] Implementation details are available at: https://github.com/eigenAyoub/embed-fusion

in online debates), SciDocs (citation and co-citation recommendation), AILAStatutes (legal domain), and QuoraRetrieval (duplicate question retrieval). Spanning biomedical, scientific, legal, and open-domain question–answer settings, and involving both short (claims, questions) and long-form (abstracts, posts) text pairs. We report Normalized Discounted Cumulative Gain at rank 10 (nDCG@10).

4 Raw concatenation

We evaluate the raw concatenation of several pairs of embedding models that vary in size and output dimensionality, as listed in Table 1. Table 2 shows that, in most cases, concatenating two models boosts performance over each participant model and occasionally surpasses much larger baselines. For example, concatenating `Arctic-m` with `bge-small` ($[M_1, S_5]$) and testing on all retrieval tasks in the MTEB benchmark[3] yields a mean score of 56.5. At the time of release, this placed the model 32nd on the (legacy) MTEB leaderboard, outperforming substantially larger models such as `gte-Qwen2-7B`[4] [7], which scored 56.24.

Model Selection. We find that concatenating models with complementary strength (performing well on different tasks) yields stronger results, suggesting that diversity across models enhances robustness. However, performance gains diminish with scale as the improvement from two to four models (C1 vs. C2) is less pronounced. We also, noticed that heavily fine-tuned models perform poorly when concatenated, or compressed with a decoder. Hence why we prefer combining base models (S_1, S_3, and S_5).

5 Effectiveness of the compression

To mitigate the linear growth in dimensionality of naive concatenation, we train (following the process in Methodology 3) a decoder to map the representation of a concatenation of small embedding models to a lower dimension, and show that it performs as good the original concatenated representation, and in some cases outperforming it. To show the effectiveness of the decoder we first evaluate it on single models and then pairs of models.

5.1 Single Model Evaluation

We show that training a decoder on the output of single models improves retrieval accuracy even when those models were originally trained with an MRL objective. Specifically, we evaluate, `Arctic-m` (denoted M_1) and `MXBAI` (T_1). As Table 3 shows, truncating to the first 384 dimensions ($M_1[:384]$ and $T_1[:384]$) drops the performance when compared to the full dimensional representations. When we instead train a decoder on the full representations (denoted $\mathrm{Dec}(M_1)$ and $\mathrm{Dec}(T_1)$) and then evaluate the decoder's first 384 dimensions ($\mathrm{Dec}(M_1)[:384]$, $\mathrm{Dec}(T_1)[:384]$), both models not only recover but improve upon their truncated baseline. These findings align with [18], but unlike their work, we target small models, and train each decoder without any task-specific corpus.

5.2 Pairwise Model Evaluation

When we train a decoder on the concatenation of pairs of models, we can in most cases recover up to 98% of the performance of the original concatenation for a target output of 384-d, regardless of the input dimensionality, as highlighted next from Table 2:

- **Dec(**$[M_1, S_5]$**)[:384]** outperforms the full representation of M_1 with only half the output dimensionality, and improves over all other 384-dimensional representations above it.
- **Dec(**$[M_1, S_5]$**)[:256]** outperforms $T_1[:256]$ using only 143M parameters, with T_1 being a 335M parameters model.
- **Dec(**$[T_1, M_1]$**)[:384]** improves over all 384-d representations, including $M_1[:384]$, $T[:384]$, and their concatenation $[T_1[:384], M_1[:384]]$. T_1 and M_1 were both trained with MRL.

5.3 Effect of the decoder's size, and MRL stops.

Following the terminology introduced in Section 3.2, we train two decoders, Dec_{512}, and Dec_{1024}, on the concatenated features $[S_3, S_5] \in \mathbb{R}^{768}$. For $\mathrm{Dec}_{512} : \mathbb{R}^{768} \to \mathbb{R}^{512}$, we use the following set of Matryoshka stops: $\mathcal{D}_{512} = \{384, 512\}$. For $\mathrm{Dec}_{1024} : \mathbb{R}^{768} \to \mathbb{R}^{1024}$, we use $\mathcal{D}_{1024} = \{32, 64, 128, 256, 384, 512, 768, 892, 1024\}$, and then truncate Dec_{1024} to its first 512 dimensions. The results presented in the last section of Table 2, highlight a general trend: decoders trained with a wide output (1024) and larger set of Matryoshka stops, then truncated to a target dimension (512) outperform decoders that are trained directly on the target dimension.

5.4 Robustness Under Extreme Compression and Quantization

While simply concatenating more embedding models yields diminishing returns, we find that it improves robustness of the output representation under extreme compression and quantization scenarios. As shown in Table 4, the two combinations C_1 (132M parameters) and C_2 (66M parameters) remain relatively competitive in retrieval performance. Applying a random projection to a high dimensional space (8192 dimensions) then 1-bit quantization to C_1 ($\mathrm{LSH}_{8192}(C_1)$), produces comparable results for both combinations, with the smaller combination C_2 even outperforming C_1 on three out of six tasks. However, in extremely compressed scenarios— such as projecting down to 1024 dimensions (a 48× compression factor for C_1) $\mathrm{LSH}_{1024}(C_1)$ outperforms $\mathrm{LSH}_{1024}(C_2)$ across all tasks.

6 Conclusion

Embedding model deployment in resource-constrained environments remains challenging due to trade-offs between performance and practicality. To address this, we concatenate multiple small embedding models and compress the resulting high-dimensional representation using a lightweight decoder trained via a Matryoshka Representation Learning objective. Our method recovers most of the performance of the original concatenated embeddings while significantly reducing dimensionality and enhancing robustness under aggressive compression and quantization. This approach offers an efficient and practical alternative to training large models from scratch.

Acknowledgments

This project has received funding from the European Union's Horizon Europe research and innovation programme under the OpenWebSearch.EU project.

[3]https://huggingface.co/PaDaS-Lab/arctic-m-bge-small
[4]https://huggingface.co/Alibaba-NLP/gte-Qwen2-7B-instruct

Table 1: Embedding models: parameter counts and output dimensions

$\mathbf{S}_i$	$\mathbf{M}_i$	$[\mathbf{M}_i,\mathbf{S}_j]$	$[\mathbf{S}_i,\mathbf{S}_j]$	$\mathbf{T}_1$
33M	109M	142M	66M	335M
384	768	1152	768	1024

Table 2: Retrieval performance of selected pairs of embedding models

	NFCorpus	SciFact	ArguAna	SciDocs	AILAStatutes	QuoraRetrieval	average
e5-small (S_1)	0.32461	0.68750	0.41794	0.17714	0.20214	0.84946	0.44313
No-Ins (S_2)	0.34921	0.72219	0.57593	0.21823	0.28411	0.88413	0.50563
gte-small (S_3)	0.34767	0.72701	0.55423	0.21394	0.24066	0.88017	0.49395
bge-small (S_5)	0.33708	0.72000	0.59499	0.19725	0.20813	0.88783	0.49088
Arctic-m (M_1)	0.36236	0.71586	0.59530	0.21492	0.28101	0.87366	0.50718
$[M_1, S_3]$	0.37305	0.73927	0.60039	0.22641	0.28166	0.88625	0.51784
Dec($[M_1, S_3]$**)[:384]**	0.36514	0.72306	0.59297	0.21870	0.27719	0.87817	0.50920
$[M_1, S_5]$	0.37561	0.73427	0.62660	0.22672	0.26633	0.89245	0.52033
Dec($[M_1, S_5]$**)[:384]**	0.37054	0.72834	0.62268	0.21998	0.27700	0.88854	0.51780
$T_1[:256]$	0.36213	0.69321	0.62297	0.21489	0.24152	0.88244	0.50286
Dec($[M_1, S_5]$**)[:256]**	0.36237	0.71579	0.61370	0.21570	0.26388	0.88290	0.50906
$[T_1[:384], M_1[:384]]$	0.37760	0.71351	0.64377	0.22154	0.28586	0.88542	0.52128
Dec$([T_1, M_1])[:384]$	0.38022	0.74014	0.64918	0.23053	0.24610	0.88641	0.52210
$[\mathbf{S}_1, \mathbf{S}_2]$ (C_2)	0.35753	0.73271	0.56407	0.22079	0.26426	0.88572	0.50418
Dec($[S_1, S_2]$**)[:384]**	0.35265	0.73345	0.55236	0.21454	0.25383	0.88350	0.49839
$[\mathbf{S}_1, \mathbf{S}_5]$	0.35594	0.72684	0.58599	0.20275	0.22882	0.88843	0.49813
Dec($[S_1, S_5]$**)[:384]**	0.34703	0.72602	0.58407	0.20171	0.24530	0.88870	0.49881
$[\mathbf{S}_3, \mathbf{S}_5]$	0.35566	0.73310	0.59653	0.21121	0.22114	0.88749	0.50086
$\mathbf{Dec}_{1024}([S_3, S_5])[:512]$	0.35207	0.73206	0.59904	0.21001	0.23302	0.88736	0.50226
$\mathbf{Dec}_{512}([S_3, S_5])$	0.35306	0.72698	0.59563	0.21152	0.23808	0.88709	0.50206

Table 3: Retrieval performance of selected single models

	NFCorpus	SciFact	ArguAna	SciDocs	AILAStatutes	QuoraRetrieval	Average
mxbai (T_1)	0.38643	0.73893	0.65467	0.23101	0.24754	0.88847	0.52451
Arctic-m (M_1)	0.36236	0.71586	0.59530	0.21492	0.28101	0.87366	0.50718
$T_1[:384]$	0.37769	0.71114	0.64241	0.22143	0.24768	0.88533	0.51428
Dec(T_1**)[:384]**	0.38078	0.73622	0.64935	0.22963	0.24218	0.88627	0.52074
$M_1[:384]$	0.35242	0.70265	0.58932	0.21050	0.29594	0.86179	0.50210
Dec(M_1**)[:384]**	0.36394	0.71668	0.59132	0.21343	0.27720	0.86608	0.50477

Table 4: Robustness of the concatenation under extreme compression ratios

	NFCorpus	SciFact	ArguAna	SciDocs	AILAStatutes	QuoraRetrieval	Average
$[S_1, S_2, S_3, S_5]$ (C_1)	**0.35983**	**0.73603**	**0.59773**	0.21912	**0.24400**	**0.88891**	**0.50760**
$[S_1, S_2]$ (C_2)	0.35753	0.73271	0.56407	**0.22079**	0.26426	0.88572	0.50418
$\mathrm{LSH}_{8192}(C_1)$	0.35284	0.71909	**0.57986**	0.21005	**0.23938**	**0.88539**	0.49777
$\mathrm{LSH}_{8192}(C_2)$	**0.35333**	**0.72860**	0.54454	**0.21362**	0.20858	0.88118	0.48831
$\mathrm{LSH}_{1024}(C_1)$	**0.32840**	**0.67087**	**0.52364**	**0.19328**	**0.24645**	**0.87240**	**0.47251**
$\mathrm{LSH}_{1024}(C_2)$	0.32218	0.66514	0.47661	0.18697	0.16807	0.86303	0.44700

GenAI Usage Disclosure

The authors used generative AI tools to assist with initial code development and minor text revisions. Specifically, AI models helped generate early versions of some code segments, which the authors carefully reviewed, modified, and adapted into the code base. Generative AI was also used for minor editorial improvements to enhance clarity and readability. The authors have verified all AI-assisted content and take full responsibility for its accuracy and integrity.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PmLR, 1597–1607.

[2] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[3] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. 604–613.

[4] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2010), 117–128.

[5] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. 2022. Matryoshka representation learning. *Advances in Neural Information Processing Systems* 35 (2022), 30233–30249.

[6] Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, et al. 2024. Gecko: Versatile text embeddings distilled from large language models. *arXiv preprint arXiv:2403.20327* (2024).

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

[8] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2020. Distilling dense representations for ranking using tightly-coupled teachers. *arXiv preprint arXiv:2010.11386* (2020).

[9] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*. 163–173.

[10] Luke Merrick, Danmei Xu, Gaurav Nuti, and Daniel Campos. 2024. Arctic-embed: Scalable, efficient, and accurate text embedding models. *arXiv preprint arXiv:2405.05374* (2024).

[11] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. MTEB: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316* (2022).

[12] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*. Springer, 232–241.

[13] Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, et al. 2024. jina-embeddings-v3: Multilingual embeddings with task lora. *arXiv preprint arXiv:2409.10173* (2024).

[14] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663* (2021).

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

[16] Shitao Xiao, Zheng Liu, Weihao Han, Jianjin Zhang, Defu Lian, Yeyun Gong, Qi Chen, Fan Yang, Hao Sun, Yingxia Shao, et al. 2022. Distill-vq: Learning retrieval oriented vector quantization by distilling knowledge from dense embeddings. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1513–1523.

[17] Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*. 641–649.

[18] Jinsung Yoon, Raj Sinha, Sercan O Arik, and Tomas Pfister. 2024. Matryoshka-Adaptor: Unsupervised and Supervised Tuning for Smaller Embedding Dimensions. *arXiv preprint arXiv:2407.20243* (2024).

[19] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Jointly optimizing query encoder and product quantization to improve retrieval performance. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2487–2496.

[20] Xuandong Zhao, Zhiguo Yu, Ming Wu, and Lei Li. 2022. Compressing sentence representation for semantic retrieval via homomorphic projective distillation. *arXiv preprint arXiv:2203.07687* (2022).