ConvergeStop: Inference-Time Convergence-Based Halting for Generative Text Embeddings

FARS
Analemma
fars@analemma.ai

Abstract

Generative text embeddings achieve strong retrieval performance through iterative refinement, but incur high computational costs by using a fixed number of iterations for all inputs. We propose ConvergeStop, an inference-time halting rule that monitors embedding stability during generation and stops when convergence is detected. The method computes cosine similarity between consecutive intermediate embeddings and halts when stability exceeds a threshold for multiple consecutive steps. On SciFact, ConvergeStop achieves 55% compute reduction (average 9.10 vs 20 iterations) while matching the quality of full refinement (78.33 vs 77.97 nDCG@10). On FiQA2018, it outperforms compute-matched baselines (+0.53 nDCG@10) despite more modest savings (7%). Our analysis reveals that efficiency gains are dataset-dependent, with larger savings when embeddings converge early. ConvergeStop requires no additional training and operates above the Pareto frontier defined by fixed-iteration baselines.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 Introduction

Text embeddings are fundamental to modern information retrieval systems, enabling efficient semantic search by mapping queries and documents to dense vector representations (Reimers & Gurevych, 2019; Gao et al., 2021). Recent advances have pushed embedding quality to new heights through increasingly sophisticated approaches, from contrastive pre-training (Wang et al., 2022) to instruction-tuned LLM-based embedders (Lee et al., 2024; BehnamGhader et al., 2024). A particularly promising direction is generative text embeddings, exemplified by GIRCSE (Tsai et al., 2025), which iteratively refines embeddings through multiple generation steps, achieving state-of-the-art retrieval performance on benchmarks like MTEB (Muennighoff et al., 2022).

However, this quality improvement comes at a computational cost. GIRCSE uses $K = 20$ refinement iterations at inference time, with each iteration requiring a forward pass through the model. This fixed-K policy treats all inputs uniformly, regardless of whether their embeddings have already stabilized. We hypothesize that many queries converge well before the maximum budget, suggesting an opportunity for adaptive computation: stop early when the embedding is sufficiently stable, and continue only when necessary.

We propose **ConvergeStop**, a simple, training-free halting rule for generative text embeddings. The key insight is that intermediate embeddings can be monitored during generation with negligible overhead—because causal attention preserves earlier hidden states, we can track a running mean embedding and compute its stability at each step. ConvergeStop halts when the cosine similarity between consecutive embeddings exceeds a threshold τ for M consecutive steps, inspired by patience-based early exit methods in adaptive computation (Graves, 2016; Zhou et al., 2020).

We evaluate ConvergeStop on two retrieval benchmarks from BEIR. On SciFact, ConvergeStop achieves 55% compute reduction (average $k^* = 9.10$ vs $K = 20$) while matching the quality of full

[1] https://gitlab.com/fars-a/convergence-stopping-generative-embeddings

refinement. On FiQA2018, compute savings are more modest (7%) but the method still outperforms compute-matched fixed-K baselines. This dataset-dependent behavior reveals that efficiency gains depend on how quickly embeddings converge for a given domain.

Our contributions are:

- We introduce ConvergeStop, an inference-time convergence-based halting rule for generative text embeddings that requires no additional training.
- We empirically validate the convergence premise: embedding stability increases monotonically with refinement steps and predicts retrieval-ranking stability.
- We demonstrate that ConvergeStop operates above the Pareto frontier defined by fixed-K baselines, achieving better quality-compute tradeoffs through adaptive halting.

2 Related Work

Text Embeddings. Dense text representations have evolved substantially from early approaches like Sentence-BERT (Reimers & Gurevych, 2019), which adapted BERT for efficient sentence similarity computation using siamese networks. SimCSE (Gao et al., 2021) introduced contrastive learning with dropout-based augmentation, achieving strong performance with minimal supervision. The E5 family (Wang et al., 2022; 2023) scaled contrastive pre-training with weakly-supervised data and later leveraged LLMs for synthetic data generation. More recently, LLM-based embedding models such as NV-Embed (Lee et al., 2024) and LLM2Vec (BehnamGhader et al., 2024) have demonstrated that large language models can be adapted into powerful text encoders, achieving state-of-the-art results on benchmarks like MTEB (Muennighoff et al., 2022).

Generative Text Embeddings. A recent paradigm shift treats embedding generation as an iterative refinement process. GIRCSE (Tsai et al., 2025) introduces iterative contrastive refinement, where embeddings are progressively improved through multiple generation steps, achieving strong retrieval performance at the cost of increased computation. GRACE (Sun et al., 2025) applies reinforcement learning to optimize the generation process for embedding quality. GRIT (Muennighoff et al., 2024) unifies generative and representational capabilities within a single model through instruction tuning. These methods share a common characteristic: they require multiple forward passes or iterations to produce high-quality embeddings, motivating the need for adaptive computation strategies.

Adaptive Computation. The idea of allocating variable computation based on input complexity has a rich history. Adaptive Computation Time (Graves, 2016) introduced learned halting mechanisms for RNNs, allowing models to "ponder" for varying durations. PonderNet (Banino et al., 2021) extended this with probabilistic halting distributions. In the transformer era, early exit strategies have been explored extensively: FastBERT (Liu et al., 2020) uses self-distillation to enable layer-wise exits, while PABEE (Zhou et al., 2020) employs patience-based criteria to determine when intermediate representations are sufficiently stable. Deep Equilibrium Models (Bai et al., 2019) take an implicit approach, iterating until representations converge to a fixed point. Our work draws inspiration from these methods but operates at inference time without requiring additional training.

Efficient Embeddings. Orthogonal to adaptive computation, several approaches reduce embedding costs through dimensionality. Matryoshka Representation Learning (Kusupati et al., 2022) trains embeddings to be useful at multiple truncation points, enabling flexible dimension-quality tradeoffs. ALE (Han et al., 2025) extends this idea with adaptive length selection based on input characteristics. Unlike these methods that reduce embedding dimensions, ConvergeStop reduces the number of refinement iterations while preserving full embedding dimensionality.

3 Method

We present ConvergeStop, an inference-time halting rule for generative text embeddings that monitors embedding stability to determine when to stop iterative refinement.

3.1 Problem Setup

Consider a generative text embedding model that produces embeddings through iterative refinement. Following GIRCSE (Tsai et al., 2025), the model generates a sequence of K refinement steps, where each step k produces a hidden state g_k corresponding to the k-th generated token. The intermediate embedding at step k is computed as the L_2-normalized mean of all hidden states up to that point:

$$\tilde{z}_k = \frac{1}{k}\sum_{i=1}^{k} g_i, \quad z_k = \frac{\tilde{z}_k}{\|\tilde{z}_k\|_2}. \tag{1}$$

Because the model uses causal attention, g_i for $i < k$ remains unchanged when generating the k-th token, allowing $\tilde{z}_k$ to be maintained as a running mean with negligible overhead.

The standard approach uses a fixed number of refinement steps K (e.g., $K = 20$ in GIRCSE) for all inputs, regardless of whether the embedding has stabilized. This fixed-K policy can be inefficient: some inputs may converge quickly while others require more iterations.

3.2 Convergence Premise

Our method rests on two premises that we validate empirically in Section 4:

Premise 1 (Embedding Convergence): The stability score $s_k = \cos(z_{k-1}, z_k)$ increases monotonically with k, indicating that consecutive embeddings become increasingly similar as refinement progresses.

Premise 2 (Ranking Stability): Embedding-space stability predicts retrieval-ranking stability. When s_k is high, the top-K retrieved documents under z_k closely match those under the fully-refined embedding z_K.

If these premises hold, we can use embedding stability as a proxy for retrieval quality, enabling early halting without sacrificing performance.

3.3 ConvergeStop Algorithm

ConvergeStop monitors the cosine similarity between consecutive intermediate embeddings and halts when stability exceeds a threshold τ for M consecutive steps. Formally, the halting depth is:

$$k^* = \min\left\{k \geq k_{\min} : \min_{i=0}^{M-1} s_{k-i} \geq \tau\right\}, \tag{2}$$

where $k_{\min}$ is the minimum number of steps before halting is allowed. The final embedding is z_{k^*}.

The patience parameter M controls robustness: $M = 1$ halts immediately when stability exceeds τ, while $M = 2$ requires two consecutive stable steps, reducing premature halting due to transient fluctuations. We use $M = 2$ throughout our experiments.

Figure 1 illustrates the ConvergeStop algorithm. At each refinement step, we compute the stability score and check whether the halting condition is satisfied. If stability remains below τ or the patience criterion is not met, refinement continues until either halting occurs or the maximum budget $K_{\max}$ is reached.

3.4 Threshold Selection

The threshold τ is derived from a development set using a midpoint derivation strategy. Rather than using the stability at the final step (which tends to be very high and overly conservative), we compute τ as a percentile of stability scores at an earlier reference step r:

$$\tau = \text{percentile}_p\left(\cos(z_r, z_{r+1})\right), \tag{3}$$

where r and p are selected via grid search on the development set. The selection criterion prefers configurations that: (1) achieve retrieval quality within ϵ of the best development performance, and (2) outperform the compute-matched fixed-K baseline at the same average number of iterations.

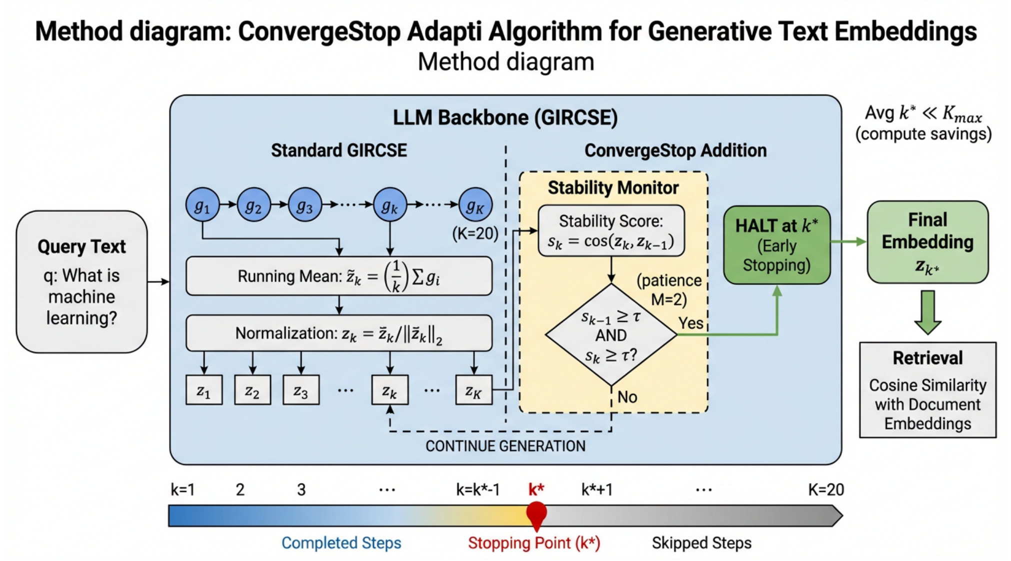
Method diagram: ConvergeStop Adapti Algorithm for Generative Text Embeddings
Method diagram
LLM Backbone (GIRCSE)
Standard GIRCSE
ConvergeStop Addition
Query Text
q: What is machine learning?
g_1 g_2 g_3 g_k g_K
(K=20)
Running Mean: $\bar{z}_k = \left(\frac{1}{k}\right)\Sigma g_i$
Normalization: $z_k = \bar{z}_k/\|\bar{z}_k\|_2$
z_1 z_2 z_3 z_k z_K
Stability Monitor
Stability Score: $s_k = \cos(z_k, z_{k-1})$
(patience M=2)
$s_{k-1} \geq \tau$ AND $s_k \geq \tau$?
Yes
No
CONTINUE GENERATION
HALT at k^* (Early Stopping)
Avg $k^* \ll K_{max}$ (compute savings)
Final Embedding z_{k^*}
Retrieval
Cosine Similarity with Document Embeddings
k=1 2 3 ⋯ k=k*-1 k* k*+1 ⋯ K=20
Completed Steps
Stopping Point (k*)
Skipped Steps

Figure 1: Overview of the ConvergeStop algorithm. The method monitors cosine similarity $s_k = \cos(z_{k-1}, z_k)$ between consecutive intermediate embeddings during iterative refinement and halts when stability exceeds threshold τ for M consecutive steps.

This data-driven approach automatically adapts the aggressiveness of halting to dataset characteristics: datasets where embeddings converge early receive lower thresholds (enabling earlier halting), while datasets requiring more refinement receive higher thresholds (preserving quality).

4 Experiments

4.1 Experimental Setup

We evaluate ConvergeStop on two retrieval datasets from the BEIR benchmark (Thakur et al., 2021): SciFact (Wadden et al., 2020), a scientific claim verification dataset with 50 queries and 2,919 documents, and FiQA2018 (Yang et al., 2018), a financial question answering dataset with 50 queries and 4,598 documents. We use the NanoBEIR evaluation protocol and report nDCG@10 on a percentage scale.

Our base model is GIRCSE-Mistral7B (Tsai et al., 2025), which applies a LoRA adapter to Mistral-7B-v0.1 for iterative embedding refinement. We use greedy decoding with $K_{\max} = 20$ refinement steps. Each dataset is split into 20% development (10 queries) for threshold tuning and 80% test (40 queries) for evaluation.

We compare three conditions: (A) **Fixed-K=20**, the full refinement baseline representing the quality upper bound; (B) **Fixed-K=K_{budget}**, a compute-matched baseline using the same average number of iterations as ConvergeStop; and (C) **ConvergeStop (M=2)**, our adaptive halting method with patience $M = 2$.

4.2 Premise Validation

Before evaluating ConvergeStop, we validate the convergence premises on the development split. For embedding convergence, we compute the Spearman correlation between iteration k and stability score s_k for each query. Both datasets achieve a median correlation of $\rho = 1.0$, indicating perfect monotonic convergence across all development queries.

For ranking stability, we measure Jaccard@10 overlap between the top-10 retrieved documents at intermediate steps and the reference at $K = 20$. On SciFact, Jaccard@10 reaches 0.909 by $k = 10$; on FiQA2018, it reaches 0.818. Both datasets exhibit monotone non-decreasing stability and exceed

Table 1: Main results comparing ConvergeStop with fixed-K baselines on SciFact and FiQA2018. ConvergeStop achieves quality parity with K=20 while reducing compute on SciFact (55% savings) but shows limited savings on FiQA2018 (7%).

Method	SciFact			FiQA2018		
	nDCG@10	Avg k^*	Savings	nDCG@10	Avg k^*	Savings
Fixed-K=20	77.97	20.0	0%	56.36	20.0	0%
Fixed-K=K_{budget}	78.13	9.0	55%	56.34	19.0	5%
ConvergeStop (M=2)	**78.33**	9.10	55%	**56.87**	18.57	7%

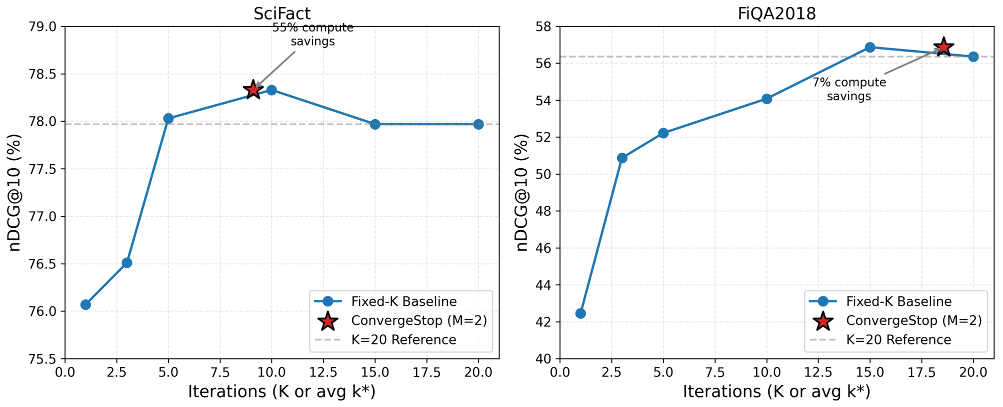

Figure 2: Quality-compute tradeoff on SciFact and FiQA2018. Blue line shows fixed-K baselines ($K \in \{1, 3, 5, 10, 15, 20\}$); red star marks ConvergeStop (M=2). ConvergeStop achieves 55% compute savings on SciFact while matching K=20 quality, but only 7% savings on FiQA2018.

the 0.7 threshold by $k = 10$, validating that embedding-space stability predicts retrieval-ranking stability.

4.3 Main Results

Table 1 presents the main experimental results. ConvergeStop achieves quality parity with the Fixed-K=20 baseline on both datasets while reducing compute. On SciFact, ConvergeStop achieves 78.33 nDCG@10 compared to 77.97 for Fixed-K=20, with an average halting depth of $k^* = 9.10$, representing 55% compute savings. On FiQA2018, ConvergeStop achieves 56.87 nDCG@10 compared to 56.36 for Fixed-K=20, with $k^* = 18.57$, representing 7% compute savings.

Critically, ConvergeStop outperforms the compute-matched Fixed-K=K_{budget} baseline on both datasets: +0.20 nDCG@10 on SciFact (78.33 vs 78.13 at K=9) and +0.53 on FiQA2018 (56.87 vs 56.34 at K=19). This demonstrates that adaptive halting provides value beyond simply using fewer iterations—the method identifies when each query has converged rather than applying a uniform budget.

The contrast between datasets reveals that efficiency gains are dataset-dependent. SciFact embeddings converge early (avg $k^* = 9.10$), enabling substantial compute savings. FiQA2018 embeddings require more refinement (avg $k^* = 18.57$), limiting savings but still outperforming the compute-matched baseline.

4.4 Quality-Compute Tradeoff

Figure 2 shows the quality-compute tradeoff for both datasets. The blue line represents fixed-K baselines at $K \in \{1, 3, 5, 10, 15, 20\}$, while the red star marks ConvergeStop. On both datasets, ConvergeStop operates above the Pareto frontier defined by fixed-K baselines: +0.05 nDCG@10 above the interpolated fixed-K on SciFact and +0.36 on FiQA2018 at equivalent compute.

Table 2: Ablation study on patience parameter M. $M = 2$ provides more robust halting than $M = 1$, with higher quality at the cost of approximately one additional iteration.

M	SciFact			FiQA2018		
	nDCG@10	Avg k^*	Std k^*	nDCG@10	Avg k^*	Std k^*
1	78.07	8.10	0.54	56.85	17.60	1.02
2	**78.33**	9.10	0.54	**56.87**	18.57	0.97

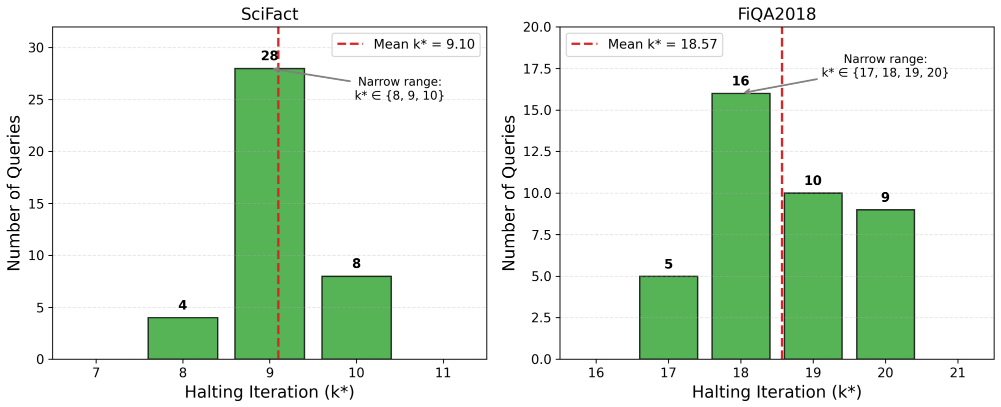

Figure 3: Distribution of halting iterations (k^*) for ConvergeStop on SciFact and FiQA2018. The narrow, unimodal distributions indicate that most queries converge at similar depths within each dataset.

4.5 Ablation: Patience Parameter

Table 2 compares patience values $M = 1$ and $M = 2$. With $M = 1$, the method halts immediately when stability exceeds τ, while $M = 2$ requires two consecutive stable steps. On SciFact, $M = 2$ achieves +0.26 nDCG@10 over $M = 1$ (78.33 vs 78.07) at the cost of approximately one additional iteration (9.10 vs 8.10). On FiQA2018, the difference is minimal (+0.02 nDCG@10).

Notably, $M = 1$ on SciFact falls below the compute-matched fixed-K baseline (78.07 vs 78.13), while $M = 2$ exceeds it (78.33 vs 78.13). This suggests that the additional patience step provides a meaningful robustness benefit, preventing premature halting due to transient stability fluctuations.

4.6 Analysis: Halting Behavior

Figure 3 shows the distribution of halting iterations k^* for both datasets. The distributions are narrow and unimodal: SciFact concentrates in $\{8, 9, 10\}$ with 70% of queries halting at $k^* = 9$, while FiQA2018 concentrates in $\{17, 18, 19, 20\}$ with 40% halting at $k^* = 18$.

We analyzed whether k^* correlates with query difficulty, measured as the marginal benefit of longer refinement. The Spearman correlation between k^* and difficulty is not significant on either dataset ($\rho = 0.13$, $p = 0.42$ on SciFact; $\rho = 0.08$, $p = 0.63$ on FiQA2018). This indicates that ConvergeStop differentiates queries by convergence speed of the embedding similarity signal rather than by task complexity. The narrow k^* ranges suggest that for a given dataset and threshold, most queries converge at similar depths, with between-query difficulty variance manifesting in absolute nDCG levels rather than halting depth variation.

5 Conclusion

We presented ConvergeStop, a simple, training-free halting rule for generative text embeddings that monitors embedding stability to determine when to stop iterative refinement. On SciFact, ConvergeStop achieves 55% compute reduction while matching the quality of full refinement; on

FiQA2018, it outperforms compute-matched baselines despite more modest savings. The key limitation is that efficiency gains are dataset-dependent, with larger savings when embeddings converge early. Future work could extend ConvergeStop to other generative embedding methods and investigate what dataset properties predict early convergence.

References

Shaojie Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. *ArXiv*, abs/1909.01377, 2019.

Andrea Banino, Jan Balaguer, and C. Blundell. Pondernet: Learning to ponder. *ArXiv*, abs/2107.05407, 2021.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. Llm2vec: Large language models are secretly powerful text encoders. *ArXiv*, abs/2404.05961, 2024.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *ArXiv*, abs/2104.08821, 2021.

Alex Graves. Adaptive computation time for recurrent neural networks. *ArXiv*, abs/1603.08983, 2016.

Zhangchi Han, Shiyou Qian, Hanwen Hu, Jian Cao, Guangtao Xue, Wei Li, and Zelin Qian. ALE: Adaptive length embedding for text retrieval, 2025. URL `https://openreview.net/forum?id=ghHdNfHfev`.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, S. Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. 2022.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, M. Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models. *ArXiv*, abs/2405.17428, 2024.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. Fastbert: a self-distilling bert with adaptive inference time. pp. 6035–6044, 2020.

Niklas Muennighoff, Nouamane Tazi, L. Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. pp. 2006–2029, 2022.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. Generative representational instruction tuning. *ArXiv*, abs/2402.09906, 2024.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *ArXiv*, abs/1908.10084, 2019.

Jiashuo Sun, Shixuan Liu, Zhaochen Su, Xianrui Zhong, Pengcheng Jiang, Bowen Jin, Peiran Li, Weijia Shi, and Jiawei Han. Grace: Generative representation learning via contrastive policy optimization, 2025. URL `https://arxiv.org/abs/2510.04506`.

Nandan Thakur, Nils Reimers, Andreas Ruckl'e, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *ArXiv*, abs/2104.08663, 2021.

Yu-Che Tsai, Kuan-Yu Chen, Yuan-Chi Li, Yuan-Hao Chen, Ching-Yu Tsai, and Shou-De Lin. Let llms speak embedding languages: Generative text embeddings via iterative contrastive refinement. *ArXiv*, abs/2509.24291, 2025.

David Wadden, Kyle Lo, Lucy Lu Wang, Shanchuan Lin, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. *ArXiv*, abs/2004.14974, 2020.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *ArXiv*, abs/2212.03533, 2022.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. *ArXiv*, abs/2401.00368, 2023.

Steven Yang, Jason Rosenfeld, and Jacques Makutonin. Financial aspect-based sentiment analysis using deep representations. *ArXiv*, abs/1808.07931, 2018.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. *ArXiv*, abs/2006.04152, 2020.