Length-Induced Embedding Collapse in PLM-based Models

Anonymous ACL submission

Abstract

Text embeddings from PLM-based models enable a wide range of applications, yet their performance often degrades on longer texts. In this paper, we introduce a phenomenon we call **Length Collapse**, where embeddings of longer texts tend to cluster together. This clustering results in a distributional inconsistency between the embeddings of short and long texts. We further investigate how these differences contribute to the performance decline observed with longer texts across various downstream tasks. Through a rigorous theoretical analysis of the self-attention mechanism, which acts as a low-pass filter in PLM-based models, we demonstrate that as text length increases, the strength of low-pass filtering intensifies, causing embeddings to retain more low-frequency components. As a result, input token features become more similar, leading to clustering and ultimately the collapse of embeddings for longer texts. To address this issue, we propose a simple method, TempScale, which mitigates the Length Collapse phenomenon. By narrowing the gap in low-pass filtering rates between long and short texts, TempScale ensures more consistent embeddings across different text lengths. This approach leads to performance improvements of **0.94%** on MTEB and **1.10%** on LongEmbed, which focuses specifically on long-context retrieval, providing strong evidence for the validity of our analysis. The source code is available at https://anonymous.4open.science/r/Length_Collapse-0FD2.

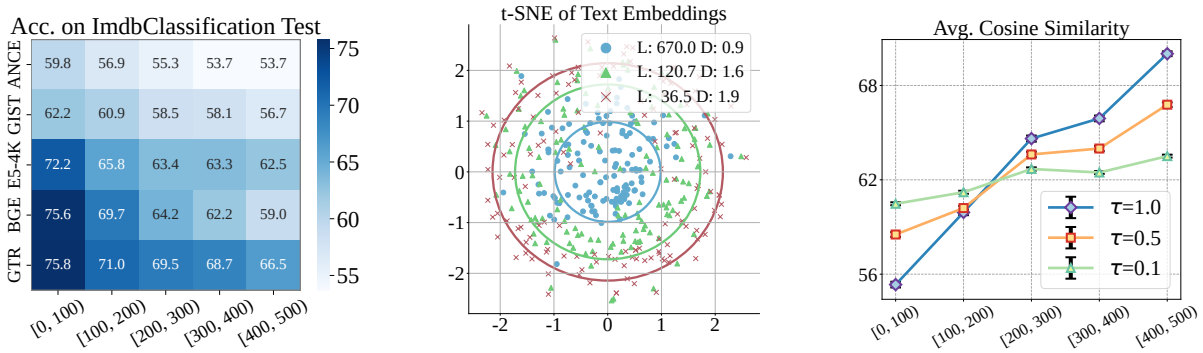
1 Introduction

Text embeddings—dense vectors that capture the semantic information of texts—are essential for many NLP applications, including text analysis (Aggarwal and Zhai, 2012; Angelov, 2020), question answering (Tan et al., 2023; Xu et al., 2024), web search (Zhao et al., 2024; Yates et al., 2021), and retrieval-augmented generation (Gao et al., 2023; Fan et al., 2024). Typically, embeddings are generated by pre-trained language models (PLMs), which produce fixed-dimensional vectors regardless of text length. In practice, we expect PLMs to perform consistently across texts of varying lengths in downstream tasks.

Unfortunately, *popular PLM-based embedding models perform poorly on longer texts*. As shown in Figure 1a, we evaluate mainstream models on the IMDB classification task from the Massive Text Embedding Benchmark (MTEB)(Muennighoff et al., 2023), using test sets grouped by text length. The results show that models with varying capabilities degrade as text length increases. For example, the BGE (Xiao et al., 2023) model's accuracy drops from 75.6% in the [0, 100) token range to 59.0% in the [400, 500) range, a 16.6% point decline.

We attribute this performance degradation to a biased behavior of embedding models: embeddings of longer texts tend to cluster together, a phenomenon we call **Length Collapse**. To verify this, we conduct controlled experiments shown in Figures 1b and 1c. Figure 1b shows that embeddings of longer texts cluster more densely near the origin in the reduced embedding space, indicating a collapse that reduces variance. Figure 1c further shows that embeddings of longer texts have higher cosine similarity to each other, leading to smaller differences. This collapse causes distributional inconsistency between embeddings of different lengths. Furthermore, we analyze how this distributional inconsistency in embeddings of different text lengths leads to performance degradation in downstream tasks such as classification, retrieval, and STS, particularly affecting the performance of longer texts.

To study how text length affects embedding distributions, we analyze the self-attention mechanism in Fourier space (Wang et al., 2022b) (§ 2.2). Building on the findings that cascading self-attention blocks act as repeated low-pass filters, we prove that the attenuation rate of this filter is propor-



(a) (b) (c)

Figure 1: **(a)** Performance of embedding models on IMDb classification across length intervals [0, 100) to [400, 500). The bluer ■ a cell, the higher the classification accuracy. **(b)** t-SNE visualization of embeddings from the BGE on NFCorpus dataset, with ● for the original dataset and ▲ and × for LLM-summarized versions, retaining semantic meaning with varying lengths. **L** indicates average text length, and **D** denotes mean distance to the origin. **(c)** Mean pairwise cosine similarity of embeddings from the BGE model across text length intervals on the corpus from NFCorpus, with an X-axis for length intervals and a Y-axis for average pairwise similarity.

tional to the largest singular value σ_a of the high-frequency components in the self-attention matrix. Assuming Gaussian-distributed input keys and query tokens, we show that σ_a decreases with increasing text length, causing longer texts to retain more of the Direct Component (DC) in token signals. This results in embeddings collapsing to a narrow space, as seen in Figure 1b.

To validate our theoretical analysis, we propose a method, Temperature Scaling (**TempScale**), to mitigate Length Collapse. TempScale manipulates the attention map by dividing the scores by a parameter called temperature (smaller than 1) before applying the $\mathrm{softmax}(\cdot)$ operator. This reduces the gap in σ_a between long and short texts in the self-attention matrix, minimizing the distributional gap between their embeddings. As shown in Figure 1c, a smaller temperature enables embeddings to exhibit lower pairwise cosine similarity, resulting in a more even distribution and alleviating Length Collapse. Additionally, TempScale improves performance by **0.94%** and **1.10%** in MTEB and LongEmbed (Zhu et al., 2024), respectively. All results confirm the validity of our analysis.

Our contributions are as follows: (1) We are the first to identify **Length Collapse**, where embeddings of longer texts cluster together, and explain how this uneven distribution causes performance degradation on long texts in downstream tasks. (2) We provide a rigorous theoretical analysis in the spectral domain, demonstrating that Length Collapse occurs due to the increasing low-pass filtering strength of self-attention as sequence length grows, causing token signals to retain only their DC component. (3) To validate this theoretical analysis, we propose a simple method, TempScale, to mitigate Length Collapse. Empirical results show a **0.94%** performance improvement across the MTEB and a **1.10%** improvement on LongEmbed, validating the correctness of our theoretical analysis.

2 Length Collapse

In this section, we define the problem and introduce our notations. We then present the Transformer structure in PLM-based models and briefly explain the Fourier transform from (Wang et al., 2022b). Using this framework, we show that the attention mechanism functions as a low-pass filter, with longer input sequences amplifying this effect, resulting in increasingly similar representations.

2.1 Preliminaries and Background

Notations. Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ denote the input feature matrix, where n is the number of tokens and d is the embedding dimension. Let $\boldsymbol{x}_i \in \mathbb{R}^d$ represent the vector for the i-th token, and $\boldsymbol{z}_j \in \mathbb{R}^n$ represent the token sequence for the j-th dimension, where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, d\}$.

Transformer Architecture. Most modern embedding models (Chen et al., 2024a; Xiong et al., 2021) use a bidirectional transformer architecture with attention mechanisms. These models typically consist of three components: an embedding layer, a stack of transformer encoder blocks with Multi-Head Self-Attention (MSA) and Feed-Forward Networks (FFN), and a pooling layer to generate the final embedding. The Self-Attention (SA) module encodes each token by aggregating information

from other tokens based on attention scores, as defined by the equation (Waswani et al., 2017):

$$\mathrm{SA}(\boldsymbol{X}) = \mathrm{softmax}\left(\frac{\boldsymbol{X}\boldsymbol{W}Q(\boldsymbol{X}\boldsymbol{W}K)^T}{\sqrt{d}}\right)\boldsymbol{X}\boldsymbol{W}_V,$$

where $\boldsymbol{W}_K$, $\boldsymbol{W}_Q$, and $\boldsymbol{W}_V$ are the key, query, and value weight matrices, with d_q and d_k as the query and key dimensions. The function $\mathrm{softmax}(\cdot)$ normalizes the attention scores. Multi-Head Self-Attention (MSA) aggregates the outputs of multiple SA heads, projected back to the hidden dimension:

$$\mathrm{MSA}(\boldsymbol{X}) = \begin{bmatrix}\mathrm{SA}_1(\boldsymbol{X}) & \cdots & \mathrm{SA}_H(\boldsymbol{X})\end{bmatrix}\boldsymbol{W}_O,$$

where H is the number of heads, and $\boldsymbol{W}_O$ projects the combined outputs to the hidden dimension.

Fourier Analysis. We use the Fourier transform as the primary analytic tool, following (Wang et al., 2022b). Let $\mathcal{F} : \mathbb{R}^n \to \mathbb{C}^n$ represent the Discrete Fourier Transform (DFT), with its inverse $\mathcal{F}^{-1} : \mathbb{C}^n \to \mathbb{R}^n$. Applying $\mathcal{F}$ to a token sequence $\boldsymbol{z}$ corresponds to multiplying by a DFT matrix, where the k-th row corresponds to the Fourier basis at frequency $\boldsymbol{f}k = [e^{2\pi j(k-1)\cdot 0}, \ldots, e^{2\pi j(k-1)\cdot(n-1)}]^\top/\sqrt{n} \in \mathbb{R}^n$, with j as the imaginary unit. Let $\tilde{\boldsymbol{z}} = \mathcal{F}\boldsymbol{z}$ denote the spectrum of $\boldsymbol{z}$, with $\tilde{\boldsymbol{z}}_{dc} \in \mathbb{C}$ and $\tilde{\boldsymbol{z}}_{hc} \in \mathbb{C}^{n-1}$ as the DC and high-frequency components, respectively. We define the Direct-Current (DC) component as $\mathcal{DC}[\boldsymbol{z}] = \tilde{\boldsymbol{z}}_{dc}\boldsymbol{f}_1 \in \mathbb{C}^n$, and the high-frequency component as $\mathcal{HC}[\boldsymbol{z}] = [\boldsymbol{f}_2, \ldots, \boldsymbol{f}n]\tilde{\boldsymbol{z}}hc \in \mathbb{C}^n$. In signal processing, a low-pass filter preserves low-frequency components while attenuating high frequencies. In this paper, we define a low-pass filter that retains only the DC component $\mathcal{DC}[\boldsymbol{z}]$, while diminishing the high-frequency components $\mathcal{HC}[\boldsymbol{z}]$. A precise definition is given in Definition 1.

Definition 1. *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be an endomorphism with f^t denoting f applied t times. The function f acts as a low-pass filter if and only if $\lim_{t\to\infty} \frac{\|\mathcal{HC}[f^t(\boldsymbol{z})]\|_2}{\|\mathcal{DC}[f^t(\boldsymbol{z})]\|_2} = 0$ for all $\boldsymbol{z} \in \mathbb{R}^n$.*

For additional details, refer to Appendix A.

2.2 Theoretical Analysis on Length Collapse

Overview. This subsection demonstrates that increasing text length n accelerates low-pass filtering in the attention matrix, making text embeddings for longer texts more similar. We justify this by analyzing self-attention's spectral-domain effect. Building on **Lemma**1, we show that the largest singular value σ_a of $\mathcal{HC}[\boldsymbol{A}]$ influences the filtering rate, with a smaller σ_a indicating greater high-frequency loss (**Theorem**2). Our analysis further reveals that longer texts result in smaller σ_a values, causing longer text embeddings to lose feature expressiveness (**Theorem**3). Consequently, we infer that longer texts yield more similar representations (**Corollary**4), leading to **Length Collapse**.

Formally, the following lemma demonstrates that the attention matrix generated by a softmax function acts as a low-pass filter, independent of the specific token features or context window.

Lemma 1. *(Attention Matrix is A Low-pass Filter) Let $\boldsymbol{A} = \mathrm{softmax}(\boldsymbol{P})$, where $\boldsymbol{P} \in \mathbb{R}^{n\times n}$. Then $\boldsymbol{A}$ must be a low-pass filter. For all $\boldsymbol{z} \in \mathbb{R}^n$,*

$$\lim_{t\to\infty} \frac{\|\mathcal{HC}[\boldsymbol{A}^t\boldsymbol{z}]\|_2}{\|\mathcal{DC}[\boldsymbol{A}^t\boldsymbol{z}]\|_2} = 0.$$

Lemma 1 follows from the Perron-Frobenius theorem (Meyer, 2000). Since all elements of the self-attention matrix are positive and each row sums to 1, the largest eigenvalue is 1. Repeated application of the self-attention matrix represents the forward process of the embedding model, and as the number of layers increases, the output retains only the DC component. A more detailed proof can be found in Theorem 1 of Wang et al. (2022b).

Understanding that self-attention matrices act as low-pass filters, we are interested in the extent to which an SA layer suppresses high-frequency components. Additionally, we provide a filter rate to illustrate the speed at which these high-frequency components are eliminated.

Theorem 2. *(Filter Rate of SA) Let σ_a be the largest singular value of $\mathcal{HC}[\boldsymbol{A}]$, and $\mathrm{SA}(\boldsymbol{X}) = \boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_V$ the self-attention output. We have:*

$$\|\mathcal{HC}[\mathrm{SA}(\boldsymbol{X})]\|_F \le \sigma_a\|\boldsymbol{W}_V\|_2\|\mathcal{HC}[\boldsymbol{X}]\|_F. \quad (1)$$

Theorem 2 suggests the high-frequency intensity ratio to the pre- and post- attention aggregation is upper bounded by $\sigma_a\|\boldsymbol{W}_V\|_2$. When $\sigma_a\|\boldsymbol{W}_V\|_2 < 1$, $\mathcal{HC}[\boldsymbol{X}]$ converges to zero exponentially. We further present Figure 6 in Appendix C.2 to justify our results, showing that the upper bound is consistent with the trend observed in the actual values. The proof of Theorem 2 can be found in Appendix B.1.

Based on the fact that a lower σ_s leads to a higher filter-pass rate, we prove in the following theorem that σ_s decreases as the input length n increases.

Theorem 3. *(Filter Rate of Different Input Length n) Let $\boldsymbol{X}\boldsymbol{W}_Q$ and $\boldsymbol{X}\boldsymbol{W}_K$ be a Gaussian matrix, where elements $q_{ij} \sim \mathcal{N}(0, \sigma_q^2)$ and $k_{ij} \sim \mathcal{N}(0, \sigma_k^2), \forall i, j$. Let $p_{ij} = \boldsymbol{q}_i^\top \boldsymbol{k}_j / \sqrt{d}$ the attention score of pair i, j, whose variance can be expressed as $\sigma_s^2 = \sigma_q^2 \sigma_k^2 + C_{cross}$, where C_{cross} is the cross-covariance of the squared queries and keys (Goodman, 1960). Then we have*

$$\sigma_a \leq \sqrt{\frac{n}{2\sqrt{1 + \frac{1}{e^{2\sigma_s^2}}}(n-1)^{\frac{3}{2}} + 1}}, \tag{2}$$

where σ_a decreases with n increasing.

The proof of Theorem 3 is in Appendix B.2. Building on the work of Fenton (1960); Nahshan et al. (2024) on log-normal variables, Theorem 3 shows that as input length n increases, σ_a decreases, suppressing high-frequency information and reducing feature expressiveness due to the self-attention matrix's low-pass filtering effect. To validate this, we sample texts of varying lengths and plot the σ_a values from the final layer's attention matrix, as shown in Figure 7 in Appendix C.3. The results confirm that σ_a decreases with text length, leading to a higher filtering rate. To facilitate further analysis, we define the temperature of the SA defined in Nahshan et al. (2024) as:

$$\tau_s = \frac{1}{\sigma_s} = \frac{1}{\sqrt{\sigma_q^2 \sigma_k^2 + C_{cross}}}. \tag{3}$$

Then denote $\tilde{p}_{ij} = p_{ij} / \sigma_s$ and each element in attention matrix $\boldsymbol{A}$ can be rewritten as follows:

$$\boldsymbol{A}_{ij} = \frac{e^{\tilde{p}_{ij}/\tau_s}}{\sum_{k=1}^{n} e^{\tilde{p}_{ik}/\tau_s}}, \tag{4}$$

where $\tilde{p}_{ij} \sim \mathcal{N}(0, 1)$ and σ_a increases with τ_s decreases. This implies that with a lower temperature τ_s, the self-attention (SA) mechanism preserves more high-frequency components in the token signals, thereby preventing collapse in long texts.

Corollary 4. *(Length Collapse in Text Embeddings) As text length n increases, the cosine similarity of their embeddings tends to increase.*

The proof of Corollary 4 in Appendix B.3 is based on the assumption that the mean of word embeddings in natural language texts remains consistent. As shown in Figure 2, we compute text embeddings by averaging word embeddings from the BGE model's embedding matrix and then assess similarity across length intervals. The results show that as text length increases, embedding similarity rises, supporting the assumption. However, we must emphasize that the increased low-pass filtering with length is the primary cause of Length Collapse. In Appendix C.4, we demonstrate that repeating two different tokens and computing the similarity between the sequences shows that the cosine similarity of their embeddings increases with length, even without any overlapping tokens.

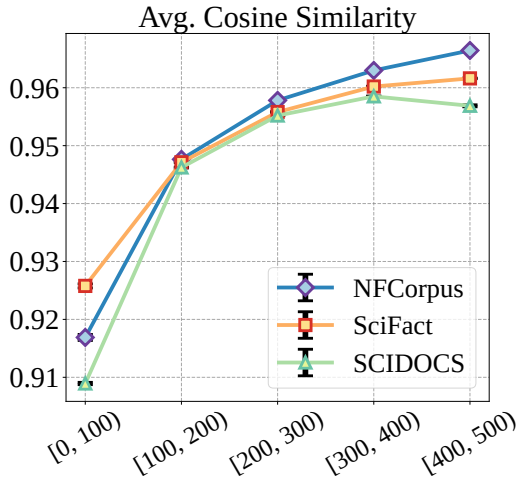


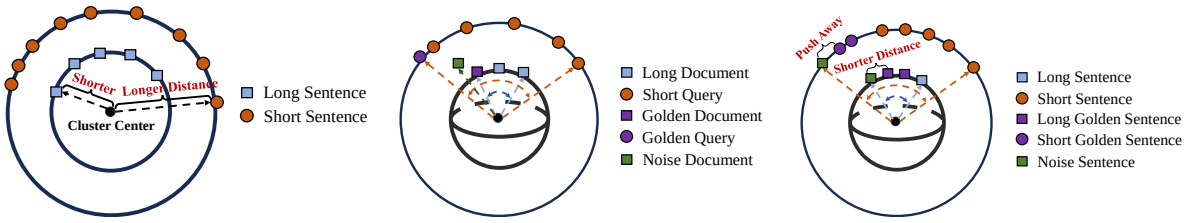
Figure 2: Mean cosine similarity of text embeddings across length intervals, with embeddings averaged from the model's word embedding matrix.

Finally, we discuss the role of other Transformer modules in Length Collapse and distinguish our work from prior research on similar collapse phenomena in Appendix C.5.

3 How does Length Collapse Lead to Performance Degradation

After analyzing how the self-attention module leads to Length Collapse, this section examines its impact on text embedding performance in the MTEB benchmark. The tasks in MTEB can be categorized into classification/clustering and matching tasks. For classification/clustering, a classifier is trained on text embeddings, while matching tasks involve calculating the similarity between embeddings using cosine similarity or dot product. Matching tasks are further divided into retrieval (for texts with significant length differences) and STS/summarization (for texts with similar lengths). We will explore how Length Collapse affects performance across these tasks.

Impact on Classification and Clustering Tasks. As shown in Figure 3 (Left), Length Collapse causes long-text embeddings to cluster at the center, while short-text embeddings spread around the



(a) Classification and Clustering (b) Retrieval (c) STS

Figure 3: A 3D toy example explains why Length Collapse leads to performance degradation.

periphery. In KNN classification, this brings clustering centers closer to long texts, resulting in a biased influence and reducing performance.

Impact on Retrieval Tasks. As shown in Figure 3 (Middle), long documents are clustered in a smaller central space, while shorter queries are more dispersed, with richer contextual representations (Ethayarajh, 2019). While centrally-positioned long documents have higher similarity with all embeddings, their representational space is more limited. This can lead to shorter noise documents (Green Square) appearing more relevant to the query (Purple Circle) due to their broader representational space. Analysis of the NFCorpus dataset, comparing rankings of the top 10% longest and shortest relevant documents (Figure 4), shows that short documents follow an inverted U-shaped distribution, whereas long documents exhibit a more uniform distribution. This is because shorter documents have a larger representational space, making it easier for them to appear at the beginning or end of the ranking list.

Impact on STS Tasks. As shown in Figure 3 (Right), Length Collapse causes long-text embeddings to cluster in a narrower space, where unrelated embeddings may show higher similarity than relevant pairs. Noise sentences (Green Square) in the long-text space may appear more similar than related sentences (Purple Square). In contrast, short-text embeddings are more spread out, resulting in better performance due to greater separation between noise and relevant sentences.

4 Mitigating Length Collapse via Temperature Scaling

As discussed in § 2.2, the self-attention matrix applies stronger low-pass filtering to longer texts, resulting in distributional differences between long and short text embeddings, which subsequently leads to a decline in long-text performance across various tasks in the MTEB benchmark. To address this issue and validate our analysis, a straightforward solution is to reduce the filtering rate difference between long and short texts, thereby mitigating the disparity in their embeddings. To achieve this, we propose a scaling technique called Temperature Scaling (**TempScale**), which directly adjusts the attention map by multiplying τ_s by a constant temperature τ less than 1, thereby reducing the difference in filtering rates σ_a between long and short texts.

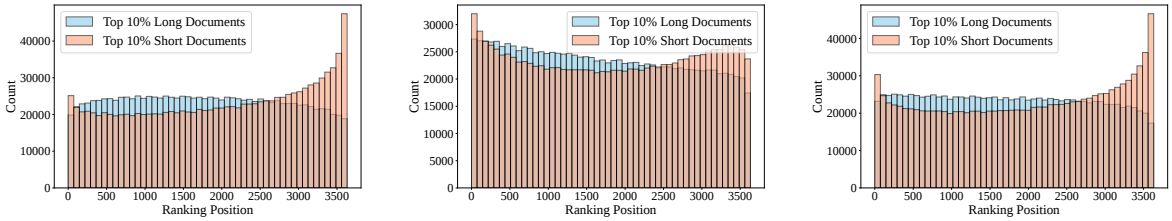
Specifically, a larger σ_s will result in a smaller factor $2\sqrt{1+\frac{1}{e^{2\sigma_s^2}}}$ in Eqn. 2, which means that as the text length n increases, σ_a will decrease more slowly, thereby reducing the difference in filtering rates between long and short texts. Therefore, by increasing σ_s, or equivalently decreasing τ_s based on Eqn. 3, we can mitigate the distributional difference between long and short text embeddings.

Formally, let $\boldsymbol{A} = \text{softmax}\left(\frac{\boldsymbol{X}\boldsymbol{W}_Q(\boldsymbol{X}\boldsymbol{W}_K)^\top}{\sqrt{d}}\right)$ denote a self-attention matrix. To decrease the τ_s of $\boldsymbol{A}$ based on Eqn. 4, we apply a temperature coefficient τ to the logits before performing the softmax operation. Specifically, for each row $\boldsymbol{p}_i$ in the attention score matrix $\frac{\boldsymbol{X}\boldsymbol{W}_Q(\boldsymbol{X}\boldsymbol{W}_K)^\top}{\sqrt{d}}$, we compute the scaled logits by dividing by a temperature $\tau \in (0, 1]$, and then apply the softmax function to obtain the attention weights:

$$\boldsymbol{A} = \text{softmax}\left(\frac{\boldsymbol{X}\boldsymbol{W}_Q(\boldsymbol{X}\boldsymbol{W}_K)^\top}{\tau\sqrt{d}}\right), \quad (5)$$

where a lower τ results a smaller τ_s.

Another Intuitive Explanation. We illustrate the effects of temperature scaling using two extreme cases to highlight how TempScale works from a different perspective. As shown in Figure 5, when scaling the matrix $\boldsymbol{A}$ with a relatively large τ, the elements of $\boldsymbol{A}$ become nearly equal, causing the matrix to filter out all high-frequency information and resulting in identical token embeddings. In contrast, when scaling with a smaller temperature, $\boldsymbol{A}$ no longer acts as a weighted sum of token representations, but selects a specific token's representation, preserving more high-frequency information. Viewing the attention matrix as an adjacency matrix, a higher temperature leads to a denser graph, enhancing information exchange between



(a) ANCE (b) GIST (c) E5-4K

Figure 4: The count of relevant documents at different ranking positions, where a smaller ranking position indicates a document is more relevant to the query.

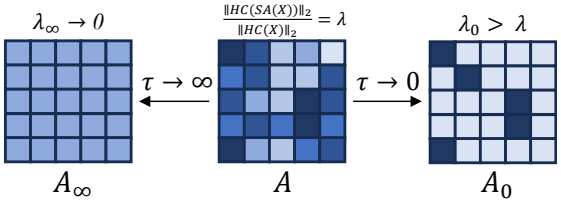


Figure 5: Two extreme cases of TempScale: larger τ causes uniform matrix elements in self-attention matrix $\boldsymbol{A}$, filtering out high-frequency information, while smaller τ preserves high-frequency details by selecting specific token representations. The darker the color, the higher the attention score.

nodes but losing high-frequency details (Oono and Suzuki, 2020; Cai and Wang, 2020). A lower temperature, on the other hand, creates a sparser graph, preserving more high-frequency information and preventing over-smoothing between nodes.

5 Experiments

In this section, we first conduct experiments to validate the effectiveness of our TempScale on MTEB and LongEmbed. Then we analyze how different tasks can benefit from TempScale to validate our theoretical analysis.

5.1 TempScale Benefits Embedding Models

Experiment Settings. We evaluate embedding models on long-context retrieval using 4 real-world tasks from LongEmbed (Zhu et al., 2024), focusing on factuality in both short-query and long-document settings. For other tasks, we select 39 datasets from MTEB (Muennighoff et al., 2023), covering classification, clustering, summarization, STS, retrieval, and reranking. Specifically, for classification, we only downloaded 10 datasets due to open-source restrictions. For retrieval, we select the 5 datasets with the longest documents to better assess Length Collapse. To comprehensively evaluate TempScale, we select several representative PLM-based embedding models, including (1) ANCE (Xiong et al., 2021); (2) GTR (Ni et al., 2022); (3) GIST (Solatorio, 2024); (4) BGE (Xiao et al., 2023); (5) E5 (Zhu et al., 2024). These models are fine-tuned from various pretrain language models, including BERT (Kenton and Toutanova, 2019), RoBERTa (Liu et al., 2019), and T5 (Chung et al., 2024). More descriptions of the datasets and models can be found in Appendix G. When evaluating the embedding models, we set the same τ on the softmax function for the attention modules across all layers within the range of $\{0.1, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. The metrics used for different tasks are consistent with MTEB and can be found in Appendix F.

Results. We select the optimal temperature τ for each model based on their performance across a single task and present the results in Table 1. The findings demonstrate that our proposed method, TempScale, improves performance across a range of general tasks, with an average increase of 0.94%. For long-context retrieval tasks, the average improvement is 1.10%. Notably, some tasks, like STS, involve texts with an average length of around 10 tokens, while datasets such as LongEmbed consist of texts exceeding 1000 tokens. TempScale proves effective for both, preventing long-text collapse and enhancing short-text embeddings, which improves downstream task performance. Additionally, larger context windows yield greater performance gains, with E5 showing the highest improvement of 1.72%. This may be due to larger windows offering more data for adjustment. Beyond the excellent performance of TempScale, more importantly, it validates the correctness of our theoretical analysis.

5.2 Further Analysis

How do the Classification and Clustering Tasks Benefit from TempScale? In § 3, we attribute the performance decline in classification and clustering tasks to the distributional differences between long and short texts, which lead the model to assign greater weight to long text embeddings during

Num. Datasets (→)	Class. 8	Clust. 11	Summ. 1	STS 10	BeirRetr. 5	Rerank. 4	LongEmbdRetr. 4	Avg. 43
				window=512				
ANCE	55.27	33.04	29.58	66.32	26.95	49.09	34.02	42.04
+TempScale	55.44	33.28	29.59	66.47	27.39	49.25	34.07	42.21
Relative Improv. (%)	0.29 ▲	0.73 ▲	0.06 ▲	0.22 ▲	1.63 ▲	0.32 ▲	0.17 ▲	0.49 ▲
GTR	55.10	38.65	29.67	70.11	33.00	54.23	37.33	45.44
+TempScale	55.59	39.52	29.83	70.26	33.56	54.22	37.33	45.76
Relative Improv. (%)	0.88 ▲	2.26 ▲	0.54 ▲	0.21 ▲	1.69 ▲	-0.01 ▼	0.01 ▲	0.80 ▲
GIST	64.75	44.77	31.14	75.61	37.55	58.55	38.21	50.08
+TempScale	65.12	44.66	32.17	75.61	37.82	58.60	38.35	50.33
Relative Improv. (%)	0.56 ▲	-0.25 ▼	3.31 ▲	-0.01 ▼	0.71 ▲	0.08 ▲	0.36 ▲	0.68 ▲
BGE	64.79	45.80	31.03	75.88	38.14	58.87	37.46	50.28
+TempScale	64.91	45.79	31.87	75.68	38.46	58.97	38.73	50.63
Relative Improv. (%)	0.19 ▲	-0.01 ▼	2.71 ▲	-0.26 ▼	0.84 ▲	0.17 ▲	3.41 ▲	1.01 ▲
				window=4k				
E5	61.72	38.82	30.58	71.77	29.77	53.12	56.01	48.83
+TempScale	62.15	40.62	31.26	72.17	30.28	53.47	56.88	49.55
Relative Improv. (%)	0.7 ▲	4.62 ▲	2.25 ▲	0.55 ▲	1.69 ▲	0.65 ▲	1.56 ▲	1.72 ▲
Avg Improv. (%)	0.53 ▲	1.47 ▲	1.77 ▲	0.14 ▲	1.31 ▲	0.24 ▲	1.10 ▲	0.94 ▲

Table 1: Average of the main metric (see Appendix F) per task on MTEB English subsets and LongEmbd. Relative Improv. means percentage increase over the performance without TempScale and improvements are highlighted with ▲ while decreasing values are denoted by ▼.

classifier training. TempScale addresses this by adjusting the longer texts to align with the same space as short texts as described in Figure 1c, ensuring that both contribute equally during training. By giving more weight to short texts during classifier training, the classification performance on short texts can also be improved. More experiments can be found in Appendix C.6.

How do Retrieval Tasks Benefit from TempScale? In § 3, we attribute the performance decline in retrieval tasks to the fact that short texts have more contextual embeddings. TempScale improves long text performance by adjusting the temperature to enhance high-frequency information, leading to a more contextual distribution. We validate this using NFCorpus and SciFact. As shown in Table 2, applying TempScale at lower temperatures improves the ranking of relevant long documents. This method reduces the bias introduced by text length. In summary, TempScale improves long document performance, with further results on additional models and datasets in Appendix C.7.

How do the STS Tasks Benefit from TempScale? TempScale enhances performance by giving long-document embeddings the same spatial representation as short texts. To verify this, we record the cosine similarity between random sentences, related sentences, and unrelated sentences at different temperatures, as shown in Table 3. The results show that as the τ in TempScale decreases, the similarity between related sentences remains relatively high, while the similarity between unrelated sentences is further reduced. Specifically, although the similarity between random sentences increases with τ in STS13 and STS14, the increase in the unrelated part is not as large as that of the random sentences. This indicates that the distance between unrelated sentences is increasing.

Can the Temperature be Set Based on the Length of the Texts? As shown in Table 2, the ranking of relevant long documents in retrieval tasks improves as the τ decreases. This suggests that using a smaller temperature τ for long texts better aligns the embeddings of long and short texts within the same distribution. For longer texts, optimal performance is achieved with a smaller τ, and further discussions can be found in Appendix C.7.

6 Discussion

LLM-based Embedding Models: While primarily focusing on PLM-based models, we also observe and discuss Length Collapse in LLM-based models in Appendix D. **Contrastive Learning:** Length Collapse increases similarity between long and short text embeddings. Although contrastive learning mitigates high similarity, performance degradation in long texts is also due to distribution differences between long and short texts, not just similarity. Details are in Appendix E.1. **Comparison with Other Methods:** We compare our

Temperature τ	1.0	0.9	0.8
ANCE	1,306.2	1,291.6	1,278.1
GTR	1,132.8	1,120.5	1,113.1
GIST	1,111.0	1,112.8	1,113.8
BGE	998.3	978.9	965.1
E5	1,193.3	1,172.3	1,162.8

(a) NFCorpus

Temperature τ	1.0	0.9	0.8
ANCE	80.7	78.7	81.5
GTR	81.7	76.6	72.4
GIST	14.4	12.4	11.3
BGE	13.3	12.3	12.4
E5	66.8	47.5	38.9

(b) SciFact

Table 2: Average ranking position with 20% longest document across different temperature τ.

Temperature	STS12			STS13			STS14		
	Random	Related	Unrelated	Random	Related	Unrelated	Random	Related	Unrelated
1.0	0.9097	0.9611	0.7973	0.8707	0.9470	0.7931	0.8877	0.9494	0.7889
0.9	0.9096	0.9612	0.7937	0.8721	0.9481	0.7942	0.8886	0.9499	0.7893
0.8	0.9097	0.9612	0.7926	0.8741	0.9490	0.7969	0.8897	0.9501	0.7912

Table 3: Average cosine similarities across different temperature settings.

method with post-processing approaches (Flow Function (Li et al., 2020), Whitening (Su et al., 2021)) and long-text methods (Chiang and Cholak (2022), YaRN (Peng et al., 2024)) in Appendix E.2 and E.3. Despite similar forms, these methods fail to address Length Collapse effectively.

7 Related Work

Text Embedding Models. Embeddings are generated by PLMs, which produce fixed-dimensional embeddings regardless of texts length, forming the basis of many NLP applications. Early word embedding models (Pennington et al., 2014) lack context awareness, while modern models (Wang et al., 2022a; Xiao et al., 2023) incorporate context through self-attention mechanisms. Transformer-based architectures like BERT (Kenton and Toutanova, 2019) and RoBERTa (Liu et al., 2019) are pre-trained on weakly supervised text pairs using contrastive loss (Gao et al., 2021) and fine-tuned on high-quality datasets. Recent work by (Muennighoff et al.) integrates generative and embedding tasks in PLM-based GritLM, enhancing performance in both areas. Among PLMs-based models (Wolf et al., 2020), self-attention is central, and this paper explores its role in Length Collapse.

Context Window Extension for Embedding Models. Despite excelling at generating vector representations, PLM-based embedding models are often limited by narrow context windows (512 tokens) (Wang et al., 2022a; Xiao et al., 2023; Ni et al., 2022), restricting their use in long-input scenarios like Wikipedia entries or meeting transcripts (Saad-Falcon et al., 2024; Zhu et al., 2024). Previous work attributes this limitation to small context windows and seeks to extend them. Current efforts to develop long-context models typically involve pre-training a backbone model from scratch with long inputs (Günther et al., 2023; Nussbaum et al., 2024; Chen et al., 2024a) or adapting existing models (Wang et al., 2024; Zhu et al., 2024), followed by training for embedding generation. However, in this paper, we discover that regardless of the model's context window size, the model consistently performs worse on longer texts than on shorter texts due to Length Collapse. Therefore, we aim to improve the performance of long texts across all context window size models by analyzing and addressing Length Collapse, rather than simply expanding context window size.

8 Conclusion

In this paper, we identify the phenomenon of Length Collapse, where the embeddings of longer texts tend to cluster together and provide a Fourier domain analysis to explain this behavior. Our theoretical findings suggest that Self-Attention inherently performs stronger low-pass filtering as the text length increases, leading to patch uniformity issues in longer sentences. Furthermore, the distributional differences between short and long text embeddings contribute to the performance decline of longer texts. To address this, we propose TempScale, which effectively balances the filtering rates for short and long texts. Our extensive experiments validate the accuracy of our analysis and the effectiveness of our method, significantly improving the performance of general embedding models on the MTEB and LongEmbed benchmarks.

Limitation

Limitation includes: **1) LLM-based embedding model:** Although we have observed a Length Collapse in LLM-based embedding models, further analysis is needed to investigate how unidirectional attention mechanisms specifically contribute to this phenomenon. Additionally, LLMs also exhibit a performance drop on long texts during text generation, which can be seen as Length Collapse. However, in this work, we only focus on PLM-based models across various tasks in MTEB. Future work could attempt to explain the reasons behind the performance decline of LLMs on long text generation from the perspective of low-pass filtering; **2) Tuning method:** The work in this paper relies on existing models and pre-trained parameters without using a training dataset. In future work, we will focus on tuning the temperature for additional improvements; **3) Analysis on more modules:** We primarily investigate the impact of the self-attention module on Length Collapse in this paper. Moving forward, we plan to explore the effects of additional modules in transformers such as LayerNorm and FFN.

References

Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text clustering algorithms. *Mining text data*, pages 77–128.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. *In* SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. * sem 2013 shared task: Semantic textual similarity. In *SEM*, pages 32–43.

Dimo Angelov. 2020. Top2vec: Distributed representations of topics. *arXiv preprint arXiv:2008.09470*.

Anil Bandhakavi, Nirmalie Wiratunga, P Deepak, and Stewart Massie. 2014. Generating a word-emotion lexicon from# emotional tweets. In *SEM)*, pages 12–21.

Ergun Biçici. 2015. Rtm-dcu: Predicting semantic similarity with referential translation machines. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 56–63.

Alexander Bondarenko, Maik Fröbe, Meriem Beloucif, Lukas Gienapp, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, et al. 2020. Overview of touché 2020: argument retrieval. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22–25, 2020, Proceedings 11*, pages 384–395. Springer.

Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A full-text learning to rank dataset for medical information retrieval. In *ECIR*, pages 716–722.

Chen Cai and Yusu Wang. 2020. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. *arXiv preprint arXiv:2003.04807*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Mingda Chen, Zewei Chu, Sam Wiseman, and Kevin Gimpel. 2021. Summscreen: A dataset for abstractive screenplay summarization. *arXiv preprint arXiv:2104.07091*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024b. Longlora: Efficient fine-tuning of long-context large language models. In *ICLR*.

David Chiang and Peter Cholak. 2022. Overcoming a theoretical limitation of self-attention. In *ACL*, pages 7654–7664.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *JMLR*, 25(70):1–53.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S Weld. 2020a. Specter: Document-level representation learning using citation-informed transformers. *arXiv preprint arXiv:2004.07180*.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. 2020b. Specter: Document-level representation learning using citation-informed transformers. In *ACL*.

Slawomir Dadas, Michał Perełkiewicz, and Rafał Poświata. 2020. Evaluation of sentence representations in Polish. In *LREC*, pages 1674–1680.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *EMNLP*. Association for Computational Linguistics.

Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. Summeval: Re-evaluating summarization evaluation. *TACL*, pages 391–409.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *SIGKDD*, pages 6491–6501.

Lawrence Fenton. 1960. The sum of log-normal probability distributions in scatter transmission systems. *IRE Transactions on communications systems*, 8(1):57–67.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, et al. 2022. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *arXiv preprint arXiv:2204.08582*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *EMNLP*, pages 6894–6910.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Gregor Geigle, Nils Reimers, Andreas Rücklé, and Iryna Gurevych. 2021. Tweac: transformer with extendable qa agent classifiers. *arXiv preprint arXiv:2104.07081*.

Leo A Goodman. 1960. On the exact variance of products. *Journal of the American statistical association*, 55(292):708–713.

Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, et al. 2023. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents. *arXiv preprint arXiv:2310.19923*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

Ziyan Jiang, Xueguang Ma, and Wenhu Chen. 2024. Longrag: Enhancing retrieval-augmented generation with long-context llms. *arXiv preprint arXiv:2406.15319*.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *TACL*, pages 317–328.

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

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *EMNLP*, pages 9119–9130.

Xueqing Liu, Chi Wang, Yue Leng, and ChengXiang Zhai. 2018. Linkso: a dataset for learning to retrieve similar question answer pairs on software development forums. In *Proceedings of the 4th ACM SIGSOFT International Workshop on NLP for Software Engineering*, pages 2–5.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv*.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *ACL*, pages 142–150.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. Www'18 open challenge: financial opinion mining and question answering. In *Companion proceedings of the the web conference 2018*, pages 1941–1942.

Carl D Meyer. 2000. *Matrix analysis and applied linear algebra*, volume 71. SIAM.

Niklas Muennighoff, SU Hongjin, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. Generative representational instruction tuning. In *ICLR 2024 Workshop: How Far Are We From AGI*.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark. In *EACL*, pages 2014–2037.

Yury Nahshan, Joseph Kampeas, and Emir Haleva. 2024. Linear log-normal attention with unbiased concentration. In *ICLR*.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, et al. 2022. Large dual encoders are generalizable retrievers. In *EMNLP*, pages 9844–9855.

Zach Nussbaum, John X Morris, Brandon Duderstadt, and Andriy Mulyar. 2024. Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*.

Kenta Oono and Taiji Suzuki. 2020. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. Yarn: Efficient context window extension of large language models. In *ICLR*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Nils Reimers, Philip Beyer, and Iryna Gurevych. 2016. Task-oriented intrinsic evaluation of semantic textual similarity. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 87–96.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP*, pages 410–420.

Jon Saad-Falcon, Daniel Y Fu, Simran Arora, Neel Guha, and Christopher Re. 2024. Benchmarking and building long-context retrieval models with loco and m2-bert. In *ICML*.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. Carer: Contextualized affect representations for emotion recognition. In *EMNLP*, pages 3687–3697.

Gizem Soğancıoğlu, Hakime Öztürk, and Arzucan Özgür. 2017. Biosses: a semantic sentence similarity estimation system for the biomedical domain. *Bioinformatics*, pages i49–i58.

Aivin V Solatorio. 2024. Gistembed: Guided in-sample selection of training negatives for text embedding fine-tuning. *arXiv preprint arXiv:2402.16829*.

Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou. 2021. Whitening sentence representations for better semantics and faster retrieval. *arXiv preprint arXiv:2103.15316*.

Yiming Tan, Dehai Min, Yu Li, Wenbo Li, Nan Hu, Yongrui Chen, and Guilin Qi. 2023. Evaluation of chatgpt as a question answering system for answering complex questions. *arXiv*.

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

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

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Peihao Wang, Wenqing Zheng, Tianlong Chen, and Zhangyang Wang. 2022b. Anti-oversmoothing in deep vision transformers via the fourier domain analysis: From theory to practice. In *ICLR*.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *UCNK*, pages 9929–9939. PMLR.

A Waswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, and I Polosukhin. 2017. Attention is all you need. In *NIPS*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020. Mind: A large-scale dataset for news recommendation. In *ACL*, pages 3597–3606.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2023. C-pack: Packaged resources to advance general chinese embedding. *arXiv preprint arXiv:2309.07597*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *ICLR*.

Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2024. Search-in-the-chain: Interactively enhancing large language models with search for knowledge-intensive tasks. In *WWW*, pages 1362–1373.

Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. 2021. Pretrained transformers for text ranking: Bert and beyond. In *WSDM*, pages 1154–1156.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *NIPS*.

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

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. 2021. Qmsum: A new benchmark for query-based multi-domain meeting summarization. *arXiv preprint arXiv:2104.05938*.

Dawei Zhu, Liang Wang, Nan Yang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. 2024. Longembed: Extending embedding models for long context retrieval. *arXiv preprint arXiv:2404.12096*.

A Background Information about Fourier Analysis

In this appendix, we provide additional background information on Fourier analysis. Specifically, consider the discrete Fourier transform (DFT) in the real-valued domain, denoted as $\mathcal{F}: \mathbb{R}^n \to \mathbb{C}^n$. The DFT can be expressed in matrix form as shown below:

$$\boldsymbol{DFT} = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{2\pi \mathrm{j}} & \cdots & e^{2\pi \mathrm{j}(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{2\pi \mathrm{j}(k-1)} & \cdots & e^{2\pi \mathrm{j}(k-1)\cdot(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{2\pi \mathrm{j}(n-1)} & \cdots & e^{2\pi \mathrm{j}(n-1)^2} \end{bmatrix}$$

and inverse discrete Fourier transform is $\boldsymbol{DFT}^{-1} = \boldsymbol{DFT}^{\top} = \boldsymbol{DFT}$. In signal processing, we can regard matrices as multi-channel signals. For example, $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ means d-channel n-length signals. When the DFT and inverse DFT are applied to multi-channel signals, each channel is transformed independently. That is, $\mathcal{F}(\boldsymbol{X}) = \begin{bmatrix} \mathcal{F}(\boldsymbol{x}_1) & \cdots & \mathcal{F}(\boldsymbol{x}_d) \end{bmatrix} = \boldsymbol{DFT} \cdot \boldsymbol{X}$.

Hereby, we can independently operators $\mathcal{DC}[\cdot]$ and $\mathcal{HC}[\cdot]$ on the echo channel using the matrices in Eqn. 6. Then we can write $\mathcal{DC}[\cdot]$ as below:

$$\begin{aligned} \mathcal{DC}[\boldsymbol{x}] &= \boldsymbol{DFT}^{-1} \operatorname{diag}(1, 0, \cdots, 0) \boldsymbol{DFT} \boldsymbol{x} \\ &= \frac{1}{n} \mathbf{1}\mathbf{1}^T \boldsymbol{x}. \end{aligned}$$

Conversely, we can denote $\mathcal{HC}[\cdot]$ as:

$$\begin{aligned} \mathcal{HC}[\boldsymbol{x}] &= \boldsymbol{DFT}^{-1} \operatorname{diag}(0, 1, \cdots, 1) \boldsymbol{DFT} \boldsymbol{x} \\ &= \boldsymbol{DFT}^{-1} (\boldsymbol{I} - \operatorname{diag}(1, 0, \cdots, 0)) \boldsymbol{DFT} \boldsymbol{x} \\ &= (\boldsymbol{I} - \frac{1}{n} \mathbf{1}\mathbf{1}^T) \boldsymbol{x}. \end{aligned}$$

B Detailed Proofs

B.1 Proof of Theorem 2

We start our analysis by providing a lemma.

Lemma 5. *The following holds* $\|\boldsymbol{AB}\|_F \leq \|\boldsymbol{A}\|_2 \|\boldsymbol{B}\|_F$ *and* $\|\boldsymbol{AB}\|_F \leq \|\boldsymbol{A}\|_F \|\boldsymbol{B}\|_2$.

Proof. Denote $\boldsymbol{B} = \begin{pmatrix} \boldsymbol{b_1} & \cdots & \boldsymbol{b_n} \end{pmatrix}$ and we have $\boldsymbol{AB} = \begin{pmatrix} \boldsymbol{Ab_1} & \cdots & \boldsymbol{Ab_n} \end{pmatrix}$. From the definition of the spectral norm, we have: $\|\boldsymbol{A}\|_2 \geq \frac{\|\boldsymbol{Ab_i}\|_2}{\|\boldsymbol{b_i}\|_2}$. Taking the average of the right-hand side, we obtain: $\|\boldsymbol{A}\|_2^2 \geq \sum_{i=1}^n \frac{\|\boldsymbol{b_i}\|_2^2}{\|\boldsymbol{B}\|_F^2} \frac{\|\boldsymbol{Ab_i}\|_2^2}{\|\boldsymbol{b_i}\|_2^2}$. This implies: $\|\boldsymbol{A}\|_2^2 \|\boldsymbol{B}\|_F^2 \geq \sum_{i=1}^n \|\boldsymbol{Ab_i}\|_2^2 = \|\boldsymbol{AB}\|_F^2$. Finally, the last step utilizes the result $\|\boldsymbol{A}\|_F^2 = \sum_{i,j} |a_{ij}|^2 = \sum_{j=1}^n \|\boldsymbol{a_j}\|_2^2$. Because both the spectral norm and the Frobenius norm of a matrix remain unchanged under transposition, we have $\|\boldsymbol{AB}\|_F = \|\boldsymbol{B}^{\top}\boldsymbol{A}^{\top}\|_F \leq \|\boldsymbol{B^T}\|_2 \|\boldsymbol{A^T}\|_F = \|\boldsymbol{A}\|_F \|\boldsymbol{B}\|_2$. □

Theorem 6. *(Filter Rate of SA) Let* σ_a *be the largest singular value of* $\mathcal{HC}[\boldsymbol{A}]$. *Define* $\mathrm{SA}(\boldsymbol{X}) = \boldsymbol{AXW}_V$ *as the output of a self-attention module, then*

$$\|\mathcal{HC}[\mathrm{SA}(\boldsymbol{X})]\|_F \leq \sigma_a \|\boldsymbol{W}_V\|_2 \|\mathcal{HC}[\boldsymbol{X}]\|_F. \quad (6)$$

Proof. First, we write $\boldsymbol{X} = \mathcal{DC}[\boldsymbol{X}] + \mathcal{HC}[\boldsymbol{X}] = \frac{1}{n}\mathbf{1}\mathbf{1}^{\top}\boldsymbol{X} + \boldsymbol{H}$, where $\boldsymbol{H} = \mathcal{HC}[\boldsymbol{X}]$ represents the remaining part of the original signals.

$$\begin{aligned} \mathcal{HC}[\mathrm{SA}(\boldsymbol{X})] &= \left(\boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right) \boldsymbol{AXW}_V && (7) \\ &= \left(\boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right) \boldsymbol{A}(\frac{1}{n}\mathbf{1}\mathbf{1}^{\top}\boldsymbol{X} + \boldsymbol{H})\boldsymbol{W}_V && (8) \\ &= \frac{1}{n}\left(\boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right) \boldsymbol{A}\mathbf{1}\mathbf{1}^{\top}\boldsymbol{XW}_V && (9) \\ &+ \left(\boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right) \boldsymbol{AHW}_V && (10) \\ &= \left(\boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right) \boldsymbol{AHW}_V && (11) \end{aligned}$$

Therefore,

$$\begin{aligned} \|\mathcal{HC}[\mathrm{SA}(\boldsymbol{X})]\|_F &= \left\| \left(\boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right) \boldsymbol{AHW}_V \right\|_F && (12) \\ &\leq \left\| \left(\boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right) \boldsymbol{A} \right\|_2 \|\boldsymbol{W}_V\|_2 \|\boldsymbol{H}\|_F && (13) \\ &= \sigma_a \|\boldsymbol{W}_V\|_2 \|\boldsymbol{H}\|_F && (14) \end{aligned}$$

The Eqn. 13 leverages inequality in Lemma 5. □

B.2 Proof of Theorem 3

Theorem 7. *(Filter Rate of Different Input Length* n*) Let* $\boldsymbol{XW}_Q$ *and* $\boldsymbol{XW}_K$ *be a Gaussian matrix, where elements* $q_{ij} \sim \mathcal{N}(0, \sigma_q^2)$ *and* $k_{ij} \sim \mathcal{N}(0, \sigma_k^2), \forall i, j$. *Let* $x_{ij} = \boldsymbol{q}_i^{\top}\boldsymbol{k}_j/\sqrt{d}$ *the attention score of pair* i, j*, whose variance can be expressed as* $\sigma_s^2 = \sigma_q^2\sigma_k^2 + C_{cross}$*, where* C_{cross} *is the cross-covariance of the squared queries and keys (Goodman, 1960). Then we have*

$$\sigma_a \leq \sqrt{\frac{n}{2\sqrt{1 + \frac{1}{e^{2\sigma_s^2}}(n-1)^{\frac{3}{2}}} + 1}}, \quad (15)$$

where σ_a *decreases with* n *increasing.*

Proof. First, we have

$$\sigma_a = \left\| \left(\boldsymbol{I} - \frac{1}{n} \mathbf{1}\mathbf{1}^T \right) \boldsymbol{A} \right\|_2 \tag{16}$$

$$\leq \left\| \boldsymbol{I} - \frac{1}{n} \mathbf{1}\mathbf{1}^T \right\|_2 \|\boldsymbol{A}\|_2 \tag{17}$$

$$\leq \|\boldsymbol{A}\|_F \tag{18}$$

The Eqn. 18 leverages $\left\| \boldsymbol{I} - \frac{1}{n} \mathbf{1}\mathbf{1}^T \right\|_2 = 1$ and $\|\boldsymbol{A}\|_2 \leq \|\boldsymbol{A}\|_F$. Now we need to upper bound $\|\boldsymbol{A}\|_F$. Generally, the product of two independent Gaussian variables has a density in the form of a modified Bessel function of the second kind (Nahshan et al., 2024). When the vector dimensions are sufficiently large, the Central Limit Theorem implies that the distribution of the dot product between $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$ can be approximated by a Gaussian distribution with zero mean and variance σ_s^2. As mentioned in Theorem 3, the variance of $\boldsymbol{q}_i^\top \boldsymbol{k}_j$ can be expressed as $\sigma^2 = \sigma_q^2 \sigma_k^2 + C_{\text{cross}}$, where $C_{\text{cross}} = \text{Cov}(\boldsymbol{q}^2, \boldsymbol{k}^2) - \text{Cov}(\boldsymbol{q}, \boldsymbol{k})^2$ is the cross-covariance of the squared queries and keys (Goodman, 1960). Thus we can suppose that each element $p_j \sim \mathcal{N}(0, \sigma_s)$ in the matrix $\boldsymbol{X}\boldsymbol{W_Q}\boldsymbol{W_K}^\top\boldsymbol{X}^\boldsymbol{T}$ is independent, where $j \in (1, \cdots, n)$:

$$\|\boldsymbol{A}\|_F = \sqrt{n \sum_{j=1}^{n} \left(\frac{e^{x_j}}{\sum_{i=1}^{n} e^{x_i}} \right)^2} \tag{19}$$

$$= \sqrt{n \frac{\sum_{j=1}^{n} e^{2x_j}}{2\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} e^{x_i + x_j} + \sum_{j=1}^{n} e^{2x_j}}} \tag{20}$$

$$= \sqrt{\frac{n e^{\ln n + 2\sigma_s^2 - \frac{1}{2}\ln\frac{e^{4\sigma_s^2}-1}{n}}}{2e^{\ln n(n-1) + \sigma_s^2 - \frac{1}{2}\ln\frac{e^{2\sigma_s^2}-1}{n(n-1)}} + e^{\ln n + 2\sigma_s^2 - \frac{1}{2}\ln\frac{e^{4\sigma_s^2}-1}{n}}}} \tag{21}$$

$$= \sqrt{\frac{n}{2\sqrt{1 + \frac{1}{e^{2\sigma_s^2}}}(n-1)^{\frac{3}{2}} + 1}}. \tag{22}$$

The derivation in Eqn. 21 primarily relies on the theorem from Fenton (1960), which addresses the sum of log-normal variables. First, we have $e^x \sim \text{LogNormal}(0, \sigma_s^2)$ and $e^{2x} \sim \text{LogNormal}(0, 4\sigma_s^2)$. Additionally, we assume that e^{x_i} and e^{x_j} are independent, leading to $e^{x_i + x_j} \sim \text{LogNormal}(0, 2\sigma_s^2)$. Now, considering the sum of log-normal variables, Fenton (1960)'s theorem provides that, for moderate values of σ^2, the sum of zero-mean i.i.d. log-normal variables can be approximated by another log-normal distribution with mean μ_Σ and variance σ_Σ^2, where:

$$\sigma_\Sigma^2 = \ln\left(\frac{1}{n}\left(e^{\sigma^2} - 1 \right) + 1 \right);$$

$$\mu_\Sigma = \ln n + (\sigma^2 - \sigma_\Sigma^2)/2.$$

For moderate values of n and σ^2, the variance σ_Σ^2 can be approximated as:

$$\sigma_\Sigma^2 \approx \ln\left(\frac{1}{n}\left(e^{\sigma^2} - 1 \right) \right).$$

Thus, the sum $\sum_{j=1}^{n} e^{2x_j}$ follows a log-normal distribution:

$$\textstyle\sum_{j=1}^{n} e^{2x_j} \sim \text{LogNormal}\left(\ln n + 2\sigma_s^2 - \frac{1}{2}\ln\left(\frac{1}{n}\left(e^{4\sigma^2} - 1 \right) \right), \ln\left(\frac{1}{n}\left(e^{4\sigma^2} - 1 \right) \right) \right).$$

Similarly, the sum $\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} e^{x_i + x_j}$ follows:

$$\textstyle\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} e^{x_i + x_j} \sim \text{LogNormal}\left(\ln n(n-1) + \sigma_s^2 - \frac{1}{2}\ln\left(\frac{e^{2\sigma_s^2}-1}{n(n-1)} \right), \ln\left(\frac{e^{2\sigma_s^2}-1}{n(n-1)} \right) \right).$$

From these, Eqn. 21 follows naturally. Finally, we have

$$\sigma_a \leq \sqrt{\frac{n}{2\sqrt{1 + \frac{1}{e^{2\sigma_s^2}}}(n-1)^{\frac{3}{2}} + 1}},$$

where σ_a decreases with n increasing. □

B.3 Proof of Corollary 4

Theorem 8. *(Length Collapse in Text Embeddings) Given two texts of length n, the cosine similarity of their text embeddings tends to increase as n grows.*

Proof. Given the two texts embeddings $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, we have

$$cos(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\left(\mathcal{HC}\left[\boldsymbol{x}_1\right] + \mathcal{DC}\left[\boldsymbol{x}_1\right]\right)\left(\mathcal{HC}\left[\boldsymbol{x}_2^T\right] + \mathcal{DC}\left[\boldsymbol{x}_2^T\right]\right)}{\|\boldsymbol{x}_1\|_2 \|\boldsymbol{x}_2\|_2} \tag{23}$$

$$= \frac{\mathcal{HC}\left[\boldsymbol{x}_1\right]\mathcal{HC}\left[\boldsymbol{x}_2^T\right] + \mathcal{DC}\left[\boldsymbol{x}_1\right]\mathcal{DC}\left[\boldsymbol{x}_2^T\right]}{\|\boldsymbol{x}_1\|_2 \|\boldsymbol{x}_2\|_2} \tag{24}$$

$$\geq \frac{\alpha^2}{\sqrt{\alpha^2 + \alpha_1^2}\sqrt{\alpha^2 + \alpha_2^2}}, \tag{25}$$

where α_1 and α_2 represent the maximum values in the frequency domain of $\mathcal{HC}\left[\boldsymbol{x}_1\right]$ and $\mathcal{HC}\left[\boldsymbol{x}_2\right]$ and α represent the value of $\mathcal{DC}\left[\boldsymbol{x}_1\right]$ and $\mathcal{DC}\left[\boldsymbol{x}_2\right]$, respectively, after applying the discrete Fourier transform. Eqn. 24 leverages that $\mathcal{HC}\left[\cdot\right]$ and $\mathcal{DC}\left[\cdot\right]$ are orthogonal. Eqn. 25 leverages that the assumption the mean of word embeddings in natural language texts maintains a relatively consistent representation. Thus $\mathcal{HC}\left[\boldsymbol{x}_1\right]$ and $\mathcal{HC}\left[\boldsymbol{x}_2\right]$ have the same value α after applying the discrete Fourier transform. Finally, according to Theorem 3, α_1 and α_2 will gradually decrease with n grows, leading to a higher cosine similarity between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. □

C More Experiments and Analysis

C.1 Rewriting Process

To investigate the differences in embedding distributions across texts of varying lengths, we use the Llama3 (*i.e.,* `Llama-3.1-8B-Instruct`) (Dubey et al., 2024) model to rewrite the texts. Specifically, we used two prompts, "*Please express the given text in one sentence. No more than 10 tokens. {{original text}}*" and "*Please use few sentences to summarize the given text. {{original text}}*", to summarize the texts. This rewriting allows the texts to retain the same semantics while having lower lengths. By studying the differences in texts of varying lengths, we conclud that the cause of the Length Collapse phenomenon is that longer texts cluster to each other in embedding space.

C.2 Details on Figure 6

To verify Theorem 2, we illustrate the high-frequency intensity of each layer's output along with its theoretical upper limit. Our visualization is based on the official checkpoint of 12-layer ANCE, GIST and BGE. We use a logarithmic scale for the purpose of a better view. Let $\boldsymbol{X}_l$ denote the output of the l-th layer. For red line, we directly calculate $\log(\|\mathcal{HC}\left[\boldsymbol{X}_{l+1}\right]\|_F / \|\mathcal{HC}\left[\boldsymbol{X}_l\right]\|_F)$ at each layer. In practice, models typically employ multi-head attention, so we replace $\|\boldsymbol{W}_V\|_2$ in Eqn. 1 with $\sigma_1 H$, where $\sigma_1 = \max_{h=1}^{H} \|\boldsymbol{W}_V^h\|_2$. For blue line, we obtain the coefficient σ_1 with respect to network parameters and apply the logarithmic scale. To summarize, Figure 6 imply an convergence rate, which is consistent with our Theorem 2. Figure **??** show the same trend as Figure 6, although the E5 model exhibits anomalies at deeper layers.

C.3 Details on Figure 7

To verify Theorem 3, we visualize the value of σ_a across different text length. Our visualization is based on the texts from NFCorpus. We sample 100 samples for each bin from 0 to 500 with a bin size of 50. The σ_a value is computed as the average of the σ_a based on the attention of all heads before the output of the final layer. To summarize, Figure 7 imply that σ_a shows a decreasing trend as n increases, which is consistent with our Theorem 3. Moreover, σ_a also increases as τ decreases, further validating our proposed method, TempScale. E5 model shows an increasing σ_a with length, which may be due to anomalies in the deep layer attention patterns.

C.4 More Analysis about Assumption in Theorem 4

In Corollary 4, we hypothesize and verify that all-natural language sequences tend to have a relatively consistent representation. As a result, different texts tend to exhibit consistent low-pass signals after losing high-frequency information, leading to ultimately consistent text embeddings. This leads to an increase in cosine similarity for longer texts. However, as shown in Figure 8, we repeat the words "dog" and "word" n times and calculate the similarity between these two texts. The results show that even when the sequences do not overlap, the text embeddings tend to converge to similar representations as the sequence length increases. This further demonstrates that Length Collapse causes completely different sequences to converge toward similarity.

C.5 More Discussion about Other Works

Other Components in Transformer. After discussing how self-attention contributes to Length Collapse, we proceed to examine the influence of other modules in the transformer, such as multi-head, residual, and FFN. Fortunately, previous work (Wang et al., 2022b) has talked about whether these components can effectively alleviate the low-pass filtering drawbacks. The proof demonstrates that while these components help preserve high-frequency signals, they do not alter the fact that the MSA block, as a whole, functions solely with the representational power of a low-pass filter. Furthermore, the ability of these models to preserve high-frequency signals is solely determined by their internal parameters and architecture, independent of the input text length. As a result, these modules do not impact our analysis of Length Collapse in practical models.

Difference from Over-Smoothing in Deeper Layers. In previous research (Wang et al., 2022b), it has been noted that a self-attention module acts as a low-pass filter, causing input feature maps to gradually lose high-frequency signals as the model layers go deeper. Furthermore, other studies (Oono and Suzuki, 2020; Cai and Wang, 2020) indicate that the node features of Graph Convolutional Networks (GCNs) can become exponentially trapped in the null space of the graph Laplacian matrix. The root cause of this phenomenon is that both graph Laplacian matrices and self-attention matrices consistently exhibit a dominant eigenvector, commonly

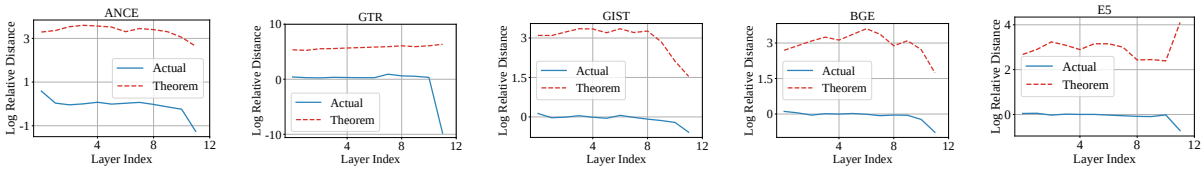
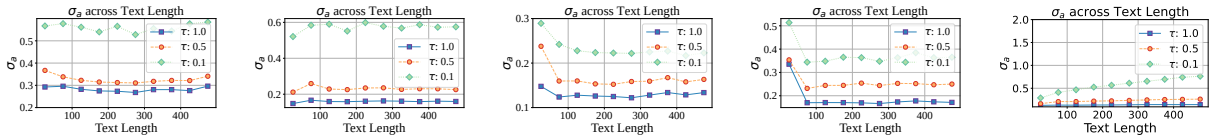


Figure 6: Visualization of the intensity of high-frequency components and their theoretical upper bounds. The blue line is defined by $\log(\|\mathcal{HC}\left[\boldsymbol{X}_{l+1}\right]\|_F / \|\mathcal{HC}\left[\boldsymbol{X}_l\right]\|_F)$, and the red line is estimated using the results in Theorem 2.



(a) ANCE (b) GTR (c) GSIT (d) BGE (e) E5

Figure 7: σ_a of $\mathcal{HC}\left[\boldsymbol{A}\right]$ before the last layer across different text length under different τ setting.

referred to as the DC component. While these studies address over-smoothing in deeper layers, we focus on how the low-pass filtering process changes as the input sequence lengthens, specifically examining over-smoothing in longer sequences.

C.6 More discussions on Classification and Clustering Tasks

For data grouping tasks like classification and clustering, in the case of N-class tasks, we can think of the model as learning N classification boundaries. The farther the text embedding is from the boundary, the closer the output probability approaches 1 or 0. As shown in Figure 9 (left), we plot the entropy of the model output probabilities across different length intervals. The model outputs higher entropy for longer texts, which may be because the embeddings of longer texts are positioned closer to the center of the space as described in Figure 1b, resulting in a shorter distance to various classification boundaries. Meanwhile, in Figure 9 (right), we can also observe that accuracy and entropy follow the same trend: the model achieves higher accuracy when it has lower entropy. In other words, the model performs better if the text embeddings are farther from the boundary. After applying TempScale, a decreased entropy will result in increased accuracy. This further supports the relationship between entropy and accuracy in classification tasks. Moreover, if the model exhibits a more severe Length Collapse phenomenon, meaning a greater performance drop on longer texts, the more performance improvement it experiences after applying TempScale. This suggests that one possible reason TempScale is effective on shorter text datasets is that it adjusts the distribution differences between texts of varying lengths. As shown in Figure 1c, after applying TempScale, the distribution of text embeddings across different length intervals becomes more uniform, which facilitates the model in learning length-agnostic parameters.

C.7 More Results on Longer Texts

Can the temperature be set based on the length of the text? In the previous experiments, we use the same temperature τ for scaling all texts in the same task. However, our analysis indicates that texts of different lengths have varying filtering rates, so a natural idea is to use different temperatures for texts of different lengths. As shown in Figure 12, We plot the performance trend for texts under the same settings in Figure 9 as the temperature varies. The results indicate that a higher temperature is optimal for short texts, while a lower temperature is preferable for long texts, as confirmed by the results in Table 2. When performing retrieval tasks, we can also set different temperatures for queries and documents to achieve better performance. As shown in Figure 11, on the QMSum dataset, we can consistently achieve better performance by setting a lower temperature for queries. Moreover, as shown in Figure 13, compared to the QMSum dataset, SummScreenFD requires a lower temperature for scaling the document due to its longer length. This further supports the conclusion that longer texts require a lower temperature for scaling. In Figure 10, we observe a similar phenomenon across other models. Except for ANCE, the performance of other models on long queries decreases as the temperature decreases. This suggests that long queries require a lower temperature to mitigate the Length Collapse phenomenon.

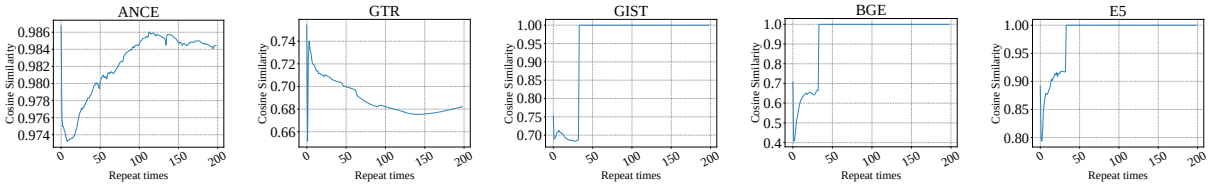


Figure 8: The cosine similarity of embeddings of texts generated by repeating words “dog” and “cat”.

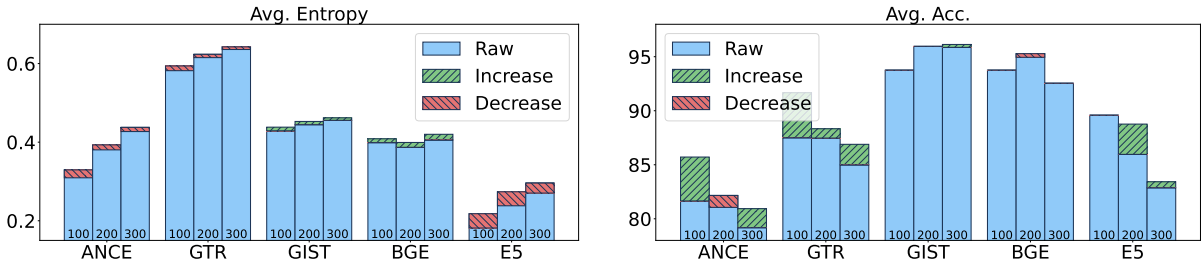


Figure 9: Probability entropy and classification accuracy of models across different length intervals on Amazon-Polarity dataset. Each model has bars representing intervals of 100 in length, with 500 text samples per interval, covering a range from 0 to 300. Bars represent raw outputs, with green and red hatching indicating increases and decreases after TempScale, respectively.

However, ANCE's performance degradation with decreasing temperature is likely attributable to its inherent limitations in processing long texts.

D Length Collapse in LLM-based Embedding Models

To explore whether Length Collapse also occurs in long-context LLMs, we select three widely used LLM-based embedding models from the MTEB benchmark—bge-multilingual-gemma2 (Chen et al., 2024a), NV-Embed-v2 (Lee et al., 2024), and e5-mistral-7b-instruct (Wang et al., 2022a)—all of which rank within the top 20 on the MTEB leaderboard. We conduct experiments using three commonly used long-text datasets, including nq and hotpotqa in LongRAG (Jiang et al., 2024) and LongAlpaca-12k (Chen et al., 2024b), which are selected based on relevance to the keyword "long" and chosen for their evenly distributed text lengths. Specifically, we analyze shifts in embedding space across different text length intervals by calculating the average Euclidean distance of each embedding from the central embedding (the mean of all embeddings) and computing cosine similarity between each pair of embeddings as in Table 4 and Table 5.

The experimental results show that as text length increases, the embeddings from different LLM-based embedding models exhibit a gradual convergence trend, with the average Euclidean distance between embeddings decreasing and pairwise cosine similarity increasing. This indicates that even mainstream long-context LLM embedding models tend to experience embedding convergence (Length Collapse) and reduced distinctiveness in long-text processing due to the low-pass filtering effect.

E More Discussions about TempScale

E.1 Relationship with Contrastive Learning

Background and Motivation. Contrastive learning is widely used to address embedding space anisotropy, reducing high similarity between random text samples of varying lengths by maximizing distances among negative pairs and aligning positive pairs. While it improves embedding quality in many applications, its impact on Length Collapse remains unexplored. This section examines how contrastive learning affects Length Collapse, assessing its contributions and limitations.

As shown in previous works (Wang and Isola, 2020), the InfoNCE loss, when scaled to a large number of negative samples, can be decomposed into two primary components: **Alignment** and **Uniformity**. Alignment ensures that positive pairs—texts with similar content—are close in the embedding space, while Uniformity spreads embeddings of negative pairs to prevent them from clustering excessively. Mathematically, as the number of negative samples $M \to \infty$, the normalized InfoNCE loss can be expressed as:

$$\lim_{M\to\infty} L(f,\tau) - \log M = -\frac{1}{\tau} E_{(x,x^+)\sim p_{\text{pos}}}\left[f(x)^T f\left(x^+\right)\right] + E_{x\sim p_{\text{data}}}\left[\log E_{x^-\sim p^-_{\text{data}}}\left[e^{f(x)^T f\left(x^-\right)/\tau}\right]\right].$$

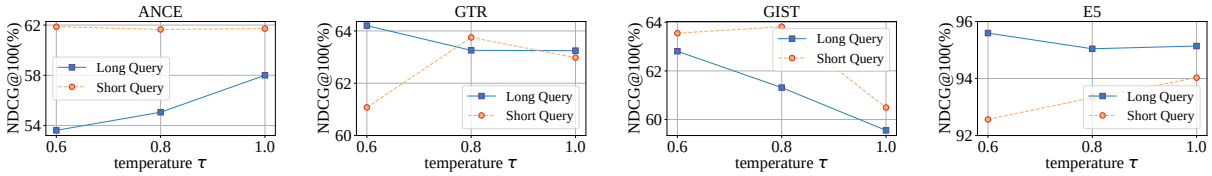


Figure 10: Results about performance difference between long query and short query across varying temperature τ on SummScreenFD dataset.

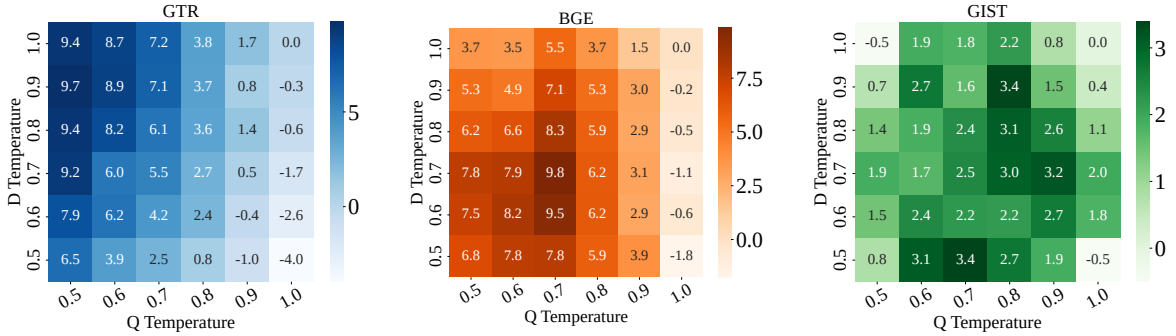


Figure 11: Relative performance compared to the raw results with varying query (Q Temperature) and document (D Temperature) temperatures using different models on QMSum.

Dataset	Model	0-1000	1000-2000	2000-3000
nq	bge-multilingual-gemma2	0.94	0.92	0.91
	NV-Embed-v2	0.98	0.91	0.89
	e5-mistral-7b-instruct	0.68	0.64	0.64
hotpotqa	bge-multilingual-gemma2	0.93	0.88	0.87
	NV-Embed-v2	0.95	0.84	0.79
	e5-mistral-7b-instruct	0.70	0.61	0.57
LongAlpaca-12k	bge-multilingual-gemma2	0.71	0.48	0.49
	NV-Embed-v2	0.89	0.86	0.87
	e5-mistral-7b-instruct	0.69	0.53	0.53

Table 4: Euclidean distance results for different datasets and LLM-based embedding models.

Dataset	Model	0-1000	1000-2000	2000-3000
nq	bge-multilingual-gemma2	0.39	0.42	0.45
	NV-Embed-v2	0.32	0.47	0.59
	e5-mistral-7b-instruct	0.75	0.77	0.78
hotpotqa	bge-multilingual-gemma2	0.51	0.52	0.59
	NV-Embed-v2	0.48	0.59	0.69
	e5-mistral-7b-instruct	0.75	0.81	0.85
LongAlpaca-12k	bge-multilingual-gemma2	0.76	0.90	0.89
	NV-Embed-v2	0.48	0.65	0.68
	e5-mistral-7b-instruct	0.78	0.87	0.87

Table 5: Pair cosine similarities results for different datasets and LLM-based embedding models.

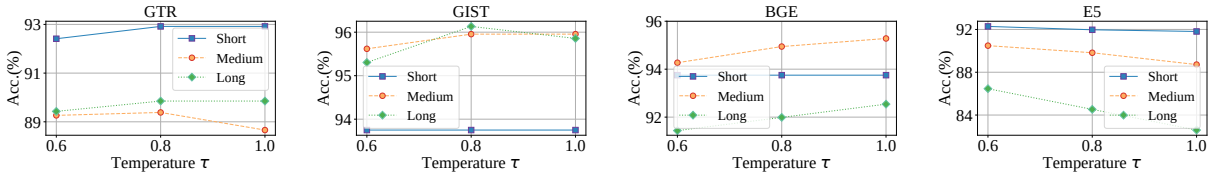


Figure 12: Results about performance difference between long query and short query across varying temperature τ on AmazonPolarity dataset.

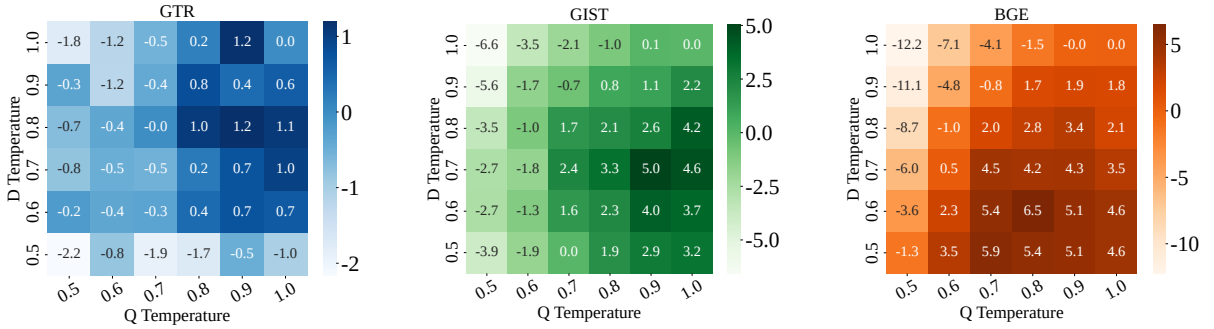


Figure 13: Relative performance compared to the raw results with varying query (Q Temperature) and document (D Temperature) temperatures using different model on SummScreenFD

These properties make contrastive learning especially effective in tasks such as retrieval, where maximizing inter-sample variance is crucial.

E.1.1 Why Contrastive Learning Cannot Fully Address Length Collapse

While contrastive learning alleviates high similarity issues across all text lengths, it does not entirely resolve the **Length Collapse** issue inherent in PLM-based models. Length Collapse arises from self-attention's tendency to push embeddings for longer texts toward a concentrated representation space. This characteristic is unaffected by contrastive learning's alignment or uniformity mechanisms because contrastive learning optimizes the relative positioning of positive and negative pairs rather than mitigating the length-induced clustering trend.

E.2 Comparison with Other Post-processing Techniques

In addition to TempScale, several post-processing methods have been proposed to address high similarity in embeddings, such as the Flow Function (Li et al., 2020) and Whitening (Su et al., 2021). For comparative analysis, we implemented the last2avg version of the Flow Function, as the full flow version requires additional training, and the Whitening method is based on its original formulation. The results, summarized in Table 6, highlight that these methods do not perform as effectively in our specific application. These results indicate that Flow and Whitening, originally developed for standard BERT embeddings, are less effective with the fine-tuned Transformer model used in this paper, which includes additional normalization layers. While TempScale reduces similarity in long-sequence embeddings, achieving distributional consistency across different sequence lengths is more critical for performance. Therefore, lowering similarity alone may not significantly improve downstream tasks.

E.3 Comparison with Other Similar Long-Text Methods

To thoroughly evaluate our TempScale approach, we compare it with two related attention scaling methods: softmax $\left(\frac{\log n}{\tau\sqrt{d}}QK^T\right)V$ (Method1) (Chiang and Cholak, 2022) and YaRN (Peng et al., 2024). In YaRN, the scaling formula for the attention matrix is given as softmax $\left(\frac{1}{\tau\sqrt{d}}QK^T\right)V$, where $\frac{1}{\tau} = 0.1\ln s + 1$, and $s = \frac{L'}{L}$, with L' representing the extended context window length and L the original context window length. These methods propose different scaling strategies, but they differ in their theoretical motivations, applications, and experimental outcomes.

While Method1, YaRN, and TempScale share a similar structural approach, they tackle different challenges. Method1 is for binary classification tasks, specifically determining if the first string in a sequence of binary strings is a '1'. It focuses on resolving straightforward decision-making problems in binary data. YaRN, by contrast, extends the context window of LLMs without requiring retrain-

Model	Rerank.	Summ.	Class.	Clust.	LongEmbdRetr.	STS	BeirRetr.	Avg.
ANCE	49.09	29.58	55.27	33.04	34.02	66.32	36.87	43.45
+Flow	48.49	29.57	56.34	32.11	31.92	66.02	36.57	43.00
+Whitening	25.16	21.76	25.82	3.21	1.29	2.78	0.79	11.54
GTR	54.23	29.67	55.10	38.65	37.33	70.11	44.98	47.15
+Flow	37.49	28.09	41.98	21.87	7.65	35.55	1.35	24.85
+Whitening	24.63	20.69	25.71	3.06	1.35	-3.02	0.63	10.44
GIST	58.55	31.14	64.75	44.77	38.21	75.61	52.77	52.26
+Flow	58.16	30.56	65.68	44.67	35.83	74.98	51.40	51.61
+Whitening	24.08	24.26	20.06	9.83	1.78	-4.31	0.81	10.93
BGE	58.87	31.03	64.79	45.80	37.46	75.88	55.29	52.73
+Flow	58.48	30.73	64.77	45.28	36.95	74.43	54.52	52.16
+Whitening	24.75	19.61	16.50	4.23	2.39	1.60	0.47	9.94
E5-4K	53.12	30.58	61.72	41.01	56.01	71.77	47.22	51.63
+Flow	52.48	29.82	62.33	39.95	44.71	68.55	30.68	46.93
+Whitening	24.94	22.14	22.73	4.04	1.85	1.96	0.56	11.17

Table 6: Performance comparison of different post-processing methods across various tasks

ing, particularly adjusting the attention mechanism to handle larger contexts. TempScale addresses Length Collapse in embedding models for long texts, improving performance without retraining by preserving distinct representations as text length increases. Moreover, Method1 offers a solution using a specially designed Transformer example, but its motivation is limited by the specificity of this example. In contrast, YaRN introduces an empirically derived scaling formula. TempScale is based on a rigorous low-pass filtering analysis, giving it a strong theoretical foundation and an intuitive explanation.

To effectively compare the performance of the above methods, we tested them on the MTEB benchmark. Since we are not modifying the context window length, we adapt YaRN as $\text{softmax}\left(\frac{\tau}{\sqrt{d}}QK^T\right)V$, with $\tau = \lambda \log n + 1$. We explore values for λ in the range $\{0.0001, 0.001, 0.01, 0.1, 1\}$. This setup is reasonable, as our findings indicate that longer texts generally benefit from a smaller temperature scale. The experimental results are as shown in Table 7.

The experimental results indicate that neither Method1 nor YaRN effectively adapts to the embedding model scenario. (Although YaRN shows slight improvement when the λ value is small, in this case, Method 2 degenerates into TempScale.) Some potential reasons are as follows: A plausible explanation is that while both methods perform finer scaling on the attention matrix, applying different temperature adjustments across varying text lengths may lead to embeddings from different lengths falling into distinct distributions, which is unfavorable for downstream tasks. Moreover, in generation tasks, Method1 and YaRN succeed likely because of differences in output requirements. For these tasks, the model outputs a probability distribution and samples from it. Minor perturbations generally don't affect the token output significantly; even if one sequence's token distribution changes, it doesn't impact the output of other sequences. In contrast, any substantial change in a single embedding for an embedding model can directly affect the entire downstream performance. For instance, in classification tasks, a classifier model relies on embeddings as input, and in retrieval tasks, a change in embedding impacts document ranking. Overall, for embedding tasks, simply applying a single temperature adjustment better maintains the overall embedding distribution, helping mitigate Length Collapse and achieve better results across various downstream applications.

F Datasets and Evaluation Metrics

Table 8 provides an overview of the datasets used in our experiments. Next, we give a brief description of the tasks involved in the experiments and the corresponding datasets and evaluation metrics they include.

Model	Rerank.	Summ.	Class.	Clust.	LongEmbdRetr.	STS	BeirRetr.	Avg.
ANCE	49.09	29.58	55.27	33.04	34.02	66.32	36.87	43.45
+Method1	41.80	29.21	48.21	20.72	6.23	53.40	7.74	29.61
+YaRN($\lambda = 0.0001$)	49.09	29.58	55.62	33.01	34.02	66.32	36.86	43.50
+YaRN($\lambda = 0.001$)	49.08	29.57	55.62	33.05	33.96	66.32	36.87	43.49
+YaRN($\lambda = 0.01$)	49.10	29.31	55.65	32.95	33.92	66.27	36.87	43.44
+YaRN($\lambda = 0.1$)	48.95	29.30	55.45	32.65	32.31	65.72	35.77	42.88
+YaRN($\lambda = 1$)	45.71	29.28	52.80	26.16	10.59	60.15	13.34	34.00
GTR	54.23	29.67	55.10	38.65	37.33	70.11	44.98	47.15
+Method1	38.12	28.44	40.34	14.46	3.61	47.09	0.82	24.70
+YaRN($\lambda = 0.0001$)	54.23	29.68	55.06	38.30	37.42	70.11	44.98	47.11
+YaRN($\lambda = 0.001$)	54.23	29.71	55.08	38.29	37.56	70.11	44.97	47.13
+YaRN($\lambda = 0.01$)	54.23	29.71	55.04	38.63	37.39	70.06	45.00	47.15
+YaRN($\lambda = 0.1$)	54.12	29.34	54.86	34.07	36.53	69.86	43.06	45.98
+YaRN($\lambda = 1$)	55.19	29.00	47.27	19.06	3.99	61.87	1.46	31.12
GIST	58.55	31.14	64.75	44.77	38.21	75.61	52.77	52.26
+Method1	58.64	28.26	51.05	28.11	5.42	62.43	7.51	34.49
+YaRN($\lambda = 0.0001$)	58.55	31.14	64.26	44.75	38.21	75.61	52.78	52.19
+YaRN($\lambda = 0.001$)	58.55	31.12	64.28	44.75	38.20	75.61	52.78	52.18
+YaRN($\lambda = 0.01$)	58.53	31.26	64.24	44.78	38.05	75.60	52.79	52.18
+YaRN($\lambda = 0.1$)	58.45	30.36	63.86	44.62	37.34	75.31	52.10	51.72
+YaRN($\lambda = 1$)	55.75	26.76	58.79	36.95	11.85	69.00	23.59	40.38
BGE	58.87	31.03	64.79	45.80	37.46	75.88	55.29	52.73
+Method1	37.78	29.17	37.80	14.36	2.10	43.72	0.84	23.68
+YaRN($\lambda = 0.0001$)	58.86	31.04	64.78	45.77	37.45	75.88	55.29	52.72
+YaRN($\lambda = 0.001$)	58.86	30.96	64.78	45.75	37.47	75.87	55.28	52.71
+YaRN($\lambda = 0.01$)	58.85	31.02	64.73	45.61	37.26	75.86	55.22	52.65
+YaRN($\lambda = 0.1$)	58.86	30.90	64.51	45.19	36.55	75.55	54.96	52.36
+YaRN($\lambda = 1$)	52.65	29.09	50.51	25.50	2.36	64.20	2.14	32.35
E5-4K	53.12	30.58	61.72	41.01	56.01	71.77	47.22	51.63
+Method1	40.90	24.11	42.85	13.64	3.24	41.62	1.05	23.92
+YaRN($\lambda = 0.0001$)	53.12	30.57	61.78	40.77	56.01	71.77	47.22	51.61
+YaRN($\lambda = 0.001$)	53.11	30.55	61.78	40.75	55.98	71.77	47.22	51.59
+YaRN($\lambda = 0.01$)	53.07	30.42	61.73	40.61	55.53	71.71	47.21	51.47
+YaRN($\lambda = 0.1$)	52.64	30.20	61.51	40.19	48.39	70.68	46.36	50.00
+YaRN($\lambda = 1$)	45.00	29.01	50.81	17.99	2.91	57.27	1.46	29.21

Table 7: Average main metric on MTEB and LongEmbd across Method1 and YaRN.

F.1 Classification

In general, we use the provided embedding model to obtain a training set and a test set. The embeddings of the training set are used to train a logistic regression classifier with a maximum of 100 iterations, which is then scored on the test set. The main evaluation metrics are accuracy, average precision, and the f1 score.

AmazonPolarity (Zhang et al., 2015) consists of Amazon customer reviews, each labeled as either "positive" or "negative."

Banking77 (Casanueva et al., 2020) dataset consists of online banking user queries labeled with one of 77 specific intents.

Emotion (Saravia et al., 2018) comprises Twitter messages categorized by six fundamental emotions: anger, fear, joy, love, sadness, and surprise.

Imdb (Maas et al., 2011) consists of extensive movie reviews categorized as either positive or negative.

MassiveIntent (FitzGerald et al., 2022) is a multilingual dataset featuring a diverse array of utterances from Amazon Alexa, each labeled with one of 60 different intents across 51 languages.

MassiveScenario (FitzGerald et al., 2022) dataset comprises a diverse collection of Amazon

Type	Name	Categ.	#Lang.	Train Samples	Dev Samples	Test Samples	Train avg. chars	Dev avg. chars	Test avg. chars
BEIR Retrival	NFCorpus	s2p	1	0	0	3,956	0	0	1,462.7
	SciFact	s2p	1	0	0	5,483	0	0	1,422.3
	SciFact	s2p	1	0	0	5,483	0	0	1,422.3
	SCIDOCS	s2p	1	0	0	26,657	0	0	1161.9
	FiQA2018	s2p	1	0	0	58,286	0	0	760.4
	Touche2020	s2p	1	0	0	382,594	0	0	1720.1
Classification	AmazonPolarityClassification	p2p	1	3,600,000	0	400,000	431.6	0	431.4
	Banking77Classification	s2s	1	10,003	0	3,080	59.5	0	54.2
	EmotionClassification	s2s	1	16,000	2,000	2,000	96.8	95.3	96.6
	ImdbClassification	p2p	1	25,000	0	25,000	1,325.1	0	1,293.8
	MassiveIntentClassification	s2s	51	11,514	2,033	2,974	35.0	34.8	34.6
	MassiveScenarioClassification	s2s	51	11,514	2,033	2,974	35.0	34.8	34.6
	ToxicConversationsClassification	s2s	1	50,000	0	50,000	298.8	0	296.6
	TweetSentimentExtractionClassification	s2s	1	27,481	0	3,534	68.3	0	67.8
Clustering	ArxivClusteringP2P	p2p	1	0	0	732,723	0	0	1,009.9
	ArxivClusteringS2S	s2s	1	0	0	732,723	0	0	74.0
	BiorxivClusteringP2P	p2p	1	0	0	75,000	0	0	1,666.2
	BiorxivClusteringS2S	s2s	1	0	0	75,000	0	0	101.6
	MedrxivClusteringP2P	p2p	1	0	0	37,500	0	0	1,981.2
	MedrxivClusteringS2S	s2s	1	0	0	37,500	0	0	114.7
	RedditClustering	s2s	1	0	420,464	420,464	0	64.7	64.7
	RedditClusteringP2P	p2p	1	0	0	459,399	0	0	727.7
	StackExchangeClustering	s2s	1	0	417,060	373,850	0	56.8	57.0
	StackExchangeClusteringP2P	p2p	1	0	0	75,000	0	0	1,090.7
	TwentyNewsgroupsClustering	s2s	1	0	0	59,545	0	0	32.0
LongEmbd Retrival	LEMBNarrativeQARetrieval	s2p	1	0	0	10,804	0	0	326,753.5
	LEMBQMSumRetrieval	s2p	1	0	0	1,724	0	0	53,335.8
	LEMBSummScreenFDRetrieval	s2p	1	0	672	0	0	30,854.3	0
	LEMBWikimQARetrieval	s2p	1	0	0	500	0	0	37,445.6
Reranking	AskUbuntuDupQuestions	s2s	1	0	0	2,255	0	0	52.5
	MindSmallReranking	s2s	1	231,530	0	107,968	69.0	0	70.9
	SciDocsRR	s2s	1	0	19,594	19,599	0	69.4	69.0
	StackOverflowDupQuestions	s2s	1	23,018	3,467	3,467	49.6	49.8	49.8
STS	BIOSSES	s2s	1	200	200	200	156.6	156.6	156.6
	SICK-R	s2s	1	19,854	19,854	19,854	46.1	46.1	46.1
	STS12	s2s	1	4,468	0	6,216	100.7	0	64.7
	STS13	s2s	1	0	0	3,000	0	0	54.0
	STS14	s2s	1	0	0	7,500	0	0	54.3
	STS15	s2s	1	0	0	6,000	0	0	57.7
	STS16	s2s	1	0	0	2,372	0	0	65.3
	STS17	s2s	11	0	0	500	0	0	43.3
	STS22	p2p	18	0	0	8,060	0	0	1,992.8
	STSBenchmark	s2s	1	11,498	3,000	2,758	57.6	64.0	53.6
Summarization	SummEval	p2p	1	0	0	2,800	0	0	359.8

Table 8: Statistics of the experimental datasets used in the work.

Alexa user utterances, each labeled with one of 60 thematic intents, and supports 51 languages.

ToxicConversations [1] , sourced from a Kaggle competition, comprises comments from the Civil Comments platform, complete with annotations indicating whether each comment is toxic.

TweetSentimentExtraction [2], a dataset from a Kaggle competition focuses on classifying tweets into three categories: neutral, positive, and negative sentiments.

[1] ToxicConversations
[2] TweetSentimentExtraction

F.2 Clustering

Clustering aims at grouping a given set of sentences or textbfs into meaningful clusters by training a mini-batch k-means model on the text embeddings. The model is scored using the v-measure (Rosenberg and Hirschberg, 2007). Since the v-measure does not depend on the cluster labels, the arrangement of the labels will not affect the score.

ArxivClusteringS2S, ArxivClusteringP2P, BiorxivClusteringS2S, BiorxivClusteringP2P, MedrxivClusteringP2P, MedrxivClusteringS2S (Muennighoff et al., 2023). These datasets are tailored for MTEB, utilizing titles or a combination of titles and abstracts from arXiv, bioRxiv, and medRxiv, with clustering labels derived from

Model Name	Publicly Available Link
ANCE	https://huggingface.co/sentence-transformers/msmarco-roberta-base-ance-firstp
GTR	https://huggingface.co/sentence-transformers/gtr-t5-base
GIST	https://huggingface.co/avsolatorio/GIST-small-Embedding-v0
BGE	https://huggingface.co/BAAI/bge-base-en-v1.5
E5	https://huggingface.co/dwzhu/e5-base-4k

Table 9: Embedding models used in the experiments.

human-assigned categories, emphasizing both main and secondary classification levels.

RedditClustering (Geigle et al., 2021), a dataset consists of titles from 199 subreddits, is organized into 25 splits, each featuring 10 to 50 classes, with every class containing between 100 and 1,000 sentences.

RedditClusteringP2P (Muennighoff et al., 2023), developed for the MTEB, consists of Reddit posts combined with their titles, organized into ten splits featuring 10 and 100 clusters each, with a total of 1,000 to 100,000 posts, aimed at clustering based on subreddit affiliation.

StackExchangeClustering (Geigle et al., 2021), a dataset consisting of titles from 121 Stack Exchange communities, is organized into 25 subsets, each containing 10 to 50 categories, with 100 to 1,000 sentences per category.

StackExchangeClusteringP2P (Muennighoff et al., 2023), designed for MTEB, comprises 10 splits of posts from StackExchange, each containing 5,000 to 10,000 entries, clustered by subreddit based on the combined content of titles and posts.

TwentyNewsgroupsClustering [3] consists of article titles from 20 different newsgroups, designed for clustering tasks, and includes 10 splits with each split featuring between 1,000 and 10,000 titles across the 20 categories.

F.3 Reranking

Reranking involves inputting a query along with a series of relevant and irrelevant reference texts, then sorting the results based on their relevance to the query. The provided model embeds the reference texts, which are compared to the query using cosine similarity. Each query is scored, and the average score across all queries is used to generate the final ranking. The evaluation metrics are MRR@k and MAP, with MAP serving as the primary metric.

AskUbuntuDupQuestions [4] dataset comprises questions sourced from AskUbuntu, accompanied by manually annotated labels that indicate whether pairs of questions are similar or dissimilar.

MindSmall (Wu et al., 2020) dataset is a comprehensive English resource designed for research in news recommendation, focusing on ranking news articles based on the title of a currently read article to suggest related content.

SciDocsRR (Cohan et al., 2020a) is a dataset designed for ranking related scientific papers using their titles as the primary basis for assessment.

StackOverflowDupQuestions (Liu et al., 2018) dataset focuses on identifying whether questions tagged with Java, JavaScript, and Python on Stack Overflow are duplicates of existing queries.

F.4 Retrieval

In retrieval task, each dataset consists of a corpus, queries, and a mapping of each query to relevant documents. The task goal is to find these relevant documents based on a given query. When evaluating, we first use the provided model to embed queries and corpus documents and then calculate the cosine similarity to obtain relevance scores and rank the corpus documents for each query based on these scores. The evaluation metrics consists of nDCG@k, MRR@k, MAP@k, precision@k, and recall@k, with nDCG@10 as the primary metric.

F.4.1 Beir Retrival

NFCorpus (Boteva et al., 2016) is a dataset that includes natural language queries sourced from NutritionFacts, paired with annotated medical documents from PubMed, utilizing the original splits from various types of content from NF, such as videos, blogs, and Q&A posts.

SciFact (Wadden et al., 2020) dataset evaluates scientific claims by matching them with evidence sourced from research literature, specifically uti-

[3] https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html

[4] https://github.com/taolei87/askubuntu

lizing a set of 300 test queries and the complete document collection from the original dataset.

SCIDOCS (Cohan et al., 2020b) is a benchmark dataset for evaluating scientific document embeddings, featuring seven tasks such as citation prediction, document classification, and recommendation.

FiQA2018 (Maia et al., 2018) is a financial question answering dataset built by crawling StackExchange posts under the Investment topic from 2009 to 2017, containing a knowledge base of 57,640 answer posts and 17,110 training question-answer pairs, with 531 testing question-answer pairs.

Touche2020 (Bondarenko et al., 2020) is a benchmark for argument retrieval, designed to support research in finding and evaluating arguments on various topics.

F.4.2 LongEmbd Retrival

LongEmbed (Zhu et al., 2024) includes 4 real-world retrieval tasks curated from long-form QA and summarization. The document in LongEmbed is much longer compared to BEIR. Thus, it can effectively evaluate the capability of the embedding model on long texts.

LEMBNarrativeQARetrieval (Kočiskỳ et al., 2018) is a question-answering dataset featuring lengthy narratives, averaging 50,474 words, that challenge models to comprehend and extract information about characters and events dispersed throughout the stories.

LEMBQMSumRetrieval (Zhong et al., 2021) dataset focuses on generating summaries of meetings based on specific queries, necessitating the extraction and synthesis of relevant information from various segments of the conversation that cover multiple topics and participants.

LEMBSummScreenFDRetrieval (Chen et al., 2021) dataset consists of pairs of transcripts from TV series and their corresponding human-crafted summaries, requiring the integration of dispersed plot elements into concise narrative descriptions.

LEMBWikimQARetrieval (Ho et al., 2020) dataset is a complex question-answering resource that includes questions requiring up to five reasoning steps, designed using specific templates to encourage deep understanding rather than simple retrieval of information.

F.5 Semantic Textual Similarity (STS)

The task goal is to determine the similarity between a pair of sentences, where continuous scores serve as labels, with higher values indicating greater similarity. The provided model embeds the sentences, and their similarity is calculated using cosine similarity. The primary evaluation metric is the Spearman correlation (Reimers et al., 2016).

STS12, STS13, STS14, STS15, STS16, STS17, STS22, STSBenchmark (Agirre et al., 2012, 2013; Bandhakavi et al., 2014; Biçici, 2015; Nakov et al., 2016) [5] [6] [7] are collections of sentence pairs designed to evaluate semantic textual similarity, with the former set focused on monolingual English pairs and the latter two incorporating cross-lingual comparisons across multiple languages.

BIOSSES (Soğancıoğlu et al., 2017) comprises 100 pairs of sentences specifically focused on the biomedical domain.

SICK-R (Dadas et al., 2020) , which stands for Sentences Involving Compositional Knowledge, comprises 100,000 diverse sentence pairs that exhibit rich lexical, syntactic, and semantic characteristics.

F.6 Summarization

The input consists of a set of summaries written by humans and machines. The goal is to score the machine-generated summaries. Use the provided model to embed the summaries. Calculate the distance between each machine summary and all human summary embeddings. Retain similar scores as the model score for each individual machine-generated summary. Calculate the Spearman correlation based on cosine similarity (Reimers et al., 2016) as the main metric.

SummEval (Fabbri et al., 2021) consists of summaries produced by advanced summarization models trained on CNN and DailyMail articles.

G Embedding Models

Table 9 provides the models used in the experiments and their publicly available links. Below is a brief introduction to these models.

ANCE (Xiong et al., 2021) enhances dense retrieval by selecting challenging negative samples from the entire corpus and asynchronously updating the Approximate Nearest Neighbor (ANN) index with each training iteration, using a context window size of 512.

[5] https://alt.qcri.org/semeval2017/task1/

[6] https://competitions.codalab.org/competitions/33835

[7] https://github.com/PhilipMay/stsb-multi-mt/

GTR (Ni et al., 2022) improves dual encoder performance for retrieval tasks by scaling up model size while keeping a fixed bottleneck embedding, leading to significant improvements in out-of-domain generalization, all within a context window size of 512.

GIST (Solatorio, 2024) consistently improves performance across different model sizes by leveraging the strengths of large, resource-intensive models to enhance smaller ones, making advanced AI technologies more accessible and cost-effective, all within a context window size of 512.

BGE (Xiao et al., 2023) offers a range of well-trained embedding models based on a BERT-like architecture, enabling users to balance performance and efficiency for various applications while also allowing easy fine-tuning. In our experiments, we use the gte-base-en-v1.5 model, which operates with a context window size of 512.

E5 (Zhu et al., 2024) is a long-context embedding model fine-tuned to support 4k token inputs while maintaining the original performance for shorter contexts, designed to advance research in long-context embedding technologies. It uses a context window size of 4k.