



國立臺灣大學電機資訊學院暨中央研究院
資料科學學位學程

碩士論文

Data Science Degree Program

College of Electrical Engineering and Computer Science

National Taiwan University and Academia Sinica

Master's Thesis

利用品質感知的多層池化方法強化文本嵌入

Enhancing Text Embeddings with Quality-Aware
Multi-Layer Pooling

蕭襄

Hsiang Hsiao

指導教授: 林守德 博士、葉彌妍 博士

Advisor: Shou-De Lin, Ph.D., Mi-Yen Yeh, Ph.D.

中華民國 114 年 8 月

August 2025

國立臺灣大學碩士學位論文
口試委員會審定書
MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

利用品質感知的多層池化方法強化文本嵌入

Enhancing Text Embeddings with Quality-Aware Multi-Layer Pooling

本論文係 蕭 襄 (R12946003) 在國立臺灣大學資料科學學位學程完成之碩士學位論文，於民國 114 年 07 月 25 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Data Science Degree Program on 25 July 2025 have examined a Master's thesis entitled above presented by HSIAO, HSIANG (R12946003) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

林守松

(指導教授 Advisor)

葉彌妍

葉彌妍 (2025年7月25日 09:16:45 PDT)

(指導教授 Advisor)

李政德

陳經懷

學位學程主管 Director:

謝宏明



誌謝

時光飛逝，兩年的研究所轉眼就到了尾聲，我也順利完成碩士論文。回頭看這一路的過程，真的有太多人要感謝。首先要感謝我的指導組守德老師和彌妍老師，在每週的會議中，守德老師總是提供許多寶貴建議，當我遇到瓶頸或實驗結果不如預期時，老師都能提出可能的問題或不同的思考方向，幫助我重新出發。彌妍老師也在過程中給予我許多指導與協助，讓我獲益良多。雖然我多次更換研究主題，但不論是什麼方向，都能感受到老師們的理解與支持，讓我能放心探索嘗試。此外，我也要感謝我的口試委員政德老師和經儂老師，在百忙之中撥空參加我的碩士口試，提出一些可以改進的地方，讓我的論文能夠更加完整與紮實。

接下來是實驗室的同學們，首先要感謝的是曾經帶我做研究的學長們昱翔、育哲、冠宇，碩一上學期主要是昱翔學長給了我很多初期的指導，幫助我建立基礎。從碩一下學期開始到畢業，則是育哲學長帶領我完成碩論，不管我遇到什麼樣的問題，有時候甚至是一些基本的觀念，育哲學長都會很有耐心地講解，直到我完全理解透徹。碩二下學期開始後有冠宇學長的加入，在這半年每週的小組討論中，冠宇學長也給予我很多指導和協助。還有要感謝曾經同組的學弟們沅錡、垣豪、清育、亮燁、泓毅，大家都在同一個大方向的研究主題上共同努力，集思廣益希望可以有好實驗結果。然後要感謝的是上一屆的學長姐們凱傑、濬睿、廷聿、威諭、庭維、昱翔、Felix，以及上兩屆的學長韋恩，在我碩一時大家都常常去實驗室，中午會一起去 118 巷吃飯聊天，有時候也會分享彼此的研究內容與



成果，其中特別感謝凱傑學長，在我剛加入實驗室和大家都還不熟時，告訴我很多實驗室的相關資訊以及幫我解決許多問題。接著要感謝的是同屆的映璇、耘安、承濬、玄鈞、豐明、子元，感謝大家在這段時間互相支持與協助，碩一的時候大家常常約吃飯聊天，還有一起修課討論作業完成分組報告等，碩二時則共同努力完成各自的碩論，了解彼此的研究內容，並給予一些有用的建議。

在就讀研究所的過程中，除了做研究之外，還有一段對我來說非常重要的經驗，就是到微軟實習，很感謝指導教授守德老師和彌妍老師的支持，讓我有機會探索學術以外的職場生活。這是我第一次進入公司實習，感謝主管 Jason、Nick 和 Mentor Yuhan、Kay、Amy 的帶領，讓我學到很多實務上的知識與技能。此外，我也很感謝其他正職 Charlie、TK、Andrew、Billy、Eva、AJ、TienYu、Denise、Gigi、Alan，上班時間中午會一起吃飯聊天，常常幫我解答工作上的各種疑難雜症之外，還不時會分享國外帶回來的伴手禮。接下來要感謝與我同期的實習生 Claire、Savina、Polo、Lewis、James、Anderlin、Eden、Enzo、Kernel、Justin、Jessie、Ruby、Camille、Laurence，在實習過程中大家互相幫忙，除了工作之外也常常約吃飯、慶生、玩桌遊，甚至在寒假一起去日本旅行，能認識這麼多好朋友，讓我的碩士生活充滿許多珍貴的快樂回憶。

最後要感謝的是家人們，因為有爸爸、媽媽、姊姊的支持與鼓勵，我才能順利完成碩士學位。即使我的求學路上經常轉換跑道，但不管我當下想走什麼方向，家人們都尊重我並支持我的選擇，所以我現在才能順利讀完研究所。每次回新竹的家時，爸爸媽媽總是會把生活上的一切都照顧得很好，讓我可以專心完成學業。姊姊也常常從花蓮到台北找我吃飯聊天，讓我在忙碌的碩士生活中可以適時地放鬆心情。在這一路上遇到的貴人很多，真的很感謝每一位在我生命出現、給予我協助和陪伴的人，我會把這份恩情銘記在心，未來也會努力成為能夠幫助他人的人，期許自己能在未來的舞台上，持續發光發熱。



摘要

文本嵌入在各種自然語言處理任務中扮演關鍵角色。近期多項研究聚焦於以大型語言模型為基礎的文本嵌入方法，並在多種下游任務中展現出卓越的效能。然而，這些方法大多僅使用模型的最後一層來提取嵌入，而有研究指出，最後一層未必能提供最佳表示。為此，本研究提出一種結合多層資訊的品質感知池化方法，根據每一層的品質指標自動分配權重，以生成更具表現力的文本嵌入。實驗結果顯示，所提方法在多數下游任務中皆能提升性能，驗證其有效性與泛化能力。

關鍵字：文本嵌入、大型語言模型、多層池化、品質感知、下游任務



Abstract

Text embeddings play a crucial role in various natural language processing tasks. Recent studies have focused on using large language models (LLMs) to generate high-quality embeddings, achieving remarkable performance on a wide range of downstream tasks. However, most existing methods rely solely on the final layer of the model to extract embeddings, despite evidence suggesting that the last layer may not always provide the most informative representation. To address this, we propose a quality-aware multi-layer pooling approach that integrates information from all layers and assigns weights based on layer-wise quality scores. Experimental results demonstrate that our method consistently improves performance across multiple downstream tasks, validating its effectiveness and generalizability.

Keywords: Text Embedding, Large Language Models, Multi-Layer Pooling, Quality-Aware, Downstream Tasks



Contents

	Page
口試委員會審定書	i
誌謝	ii
摘要	iv
Abstract	v
Contents	vi
List of Figures	viii
List of Tables	ix
Chapter 1 Introduction	1
Chapter 2 Related Work	6
2.1 Training-Based Methods	6
2.2 Training-Free Methods	7
Chapter 3 Methodology	8
3.1 Problem Definition	8
3.2 Quality Metric Design for Text Embeddings	9
3.3 Bayesian Optimization with TPE	10
Chapter 4 Experiment	13
4.1 Experiment Setup	13

4.1.1	Datasets	13
4.1.2	Training-Free Methods	13
4.1.3	Base Models	14
4.1.4	Baselines	15
4.1.5	Statistical Test	15
4.2	Main Results	15
4.3	The Upper Bound of Task-Agnostic Optimization	19
Chapter 5	Ablation Study	21
5.1	Comparison of Different Objective Function	21
5.2	Comparison of Different Samplers	22
5.3	Comparison of Different Trial Numbers	23
Chapter 6	Conclusion	25
	References	26





List of Figures

- 1.1 Last layer does not always yield best performance (the STAR sign stands for the best layer). For clearer visualization in the figure, the downstream performance of each line is normalized to a range between 0 and 1. 3



List of Tables

2.1	Categorization of existing text embedding models by LLM dependency and training supervision.	6
4.1	Statistics of datasets.	14
4.2	Experimental results on DeepSeek-R1-Distill-Qwen-7B and Mistral-7B. .	16
4.3	Experimental results on Qwen2-7B and LLaMA3.1-8B.	17
4.4	Upper bound estimation using downstream-aware optimization.	20
5.1	Downstream performance optimized by various quality metrics.	22
5.2	Correlations between quality metric scores and downstream performance.	22
5.3	Comparative results of different sampling strategies.	23
5.4	Performance comparison across different numbers of optimization trials. .	24



Chapter 1 Introduction

Text embeddings serve as a universal representation of textual data by mapping input text into dense, continuous vector spaces. These embeddings are foundational for a wide range of downstream tasks, such as text classification [20], information retrieval [16, 25], and semantic textual similarity [3, 8]. While various models exist to generate embeddings, large language models (LLMs) have recently gained traction as powerful text encoders.

LLMs can be used for text embedding through two main approaches: training-based [6, 23, 26, 36] and training-free [18, 22, 31, 34, 37] methods. The training-based approach, such as contrastive learning, encourages semantically similar texts to be closer in the embedding space while pushing apart dissimilar ones. However, this approach typically requires large-scale labeled data and considerable training time. In contrast, training-free methods leverage prompts to directly extract embeddings from pretrained LLMs. These methods eliminate the need for additional training and are therefore more efficient and scalable. In this work, we focus on training-free methods due to their practicality and flexibility.

Despite their effectiveness, existing text embedding models share a common limitation: they typically rely on the hidden states from the last layer of the model, regardless of the pooling strategy used (e.g., mean pooling [6, 26], last token pooling [36], or trainable

pooling layers [21]). However, prior studies [22, 29, 33] suggest that the last layer does not always yield the best performance. Different layers encode different types of information, and intermediate layers may capture useful semantics for downstream tasks that the last layer overlooks.

To verify the effectiveness of using the last layer to construct an embedding, we analyze the layer-wise downstream performance across models and datasets, as illustrated in Figure 1.1. The upper part fixes a dataset and plots the downstream performance of each layer across different models, while the lower part fixes a model and shows the downstream performance of each layer across different datasets. Several key observations can be made from the figure. First, regardless of the setting, the last layer is usually not the one that yields the best performance. Moreover, for a fixed dataset, the optimal layer for generating embeddings varies across different models. Conversely, for a fixed model, the optimal layer of each dataset also varies. These findings suggest that hidden states from layers other than the last may better support different downstream tasks.

Motivated by these observations, the primary research question of this work is: Can the embeddings from different layers be combined to create a new representation that enhances downstream performance? In this paper we focus on finding a set of weights to linearly combine the embeddings of different layers to generate a better representation.

The major challenge lies in the fact that the optimal weights often depend on the downstream tasks or datasets, which are most likely unknown during the process of embedding generation. Since the downstream datasets are unknown, we cannot directly optimize the weights based on the downstream performance. Another key novelty of this work lies in a generalizable and efficient means to estimate the embedding quality for optimization,

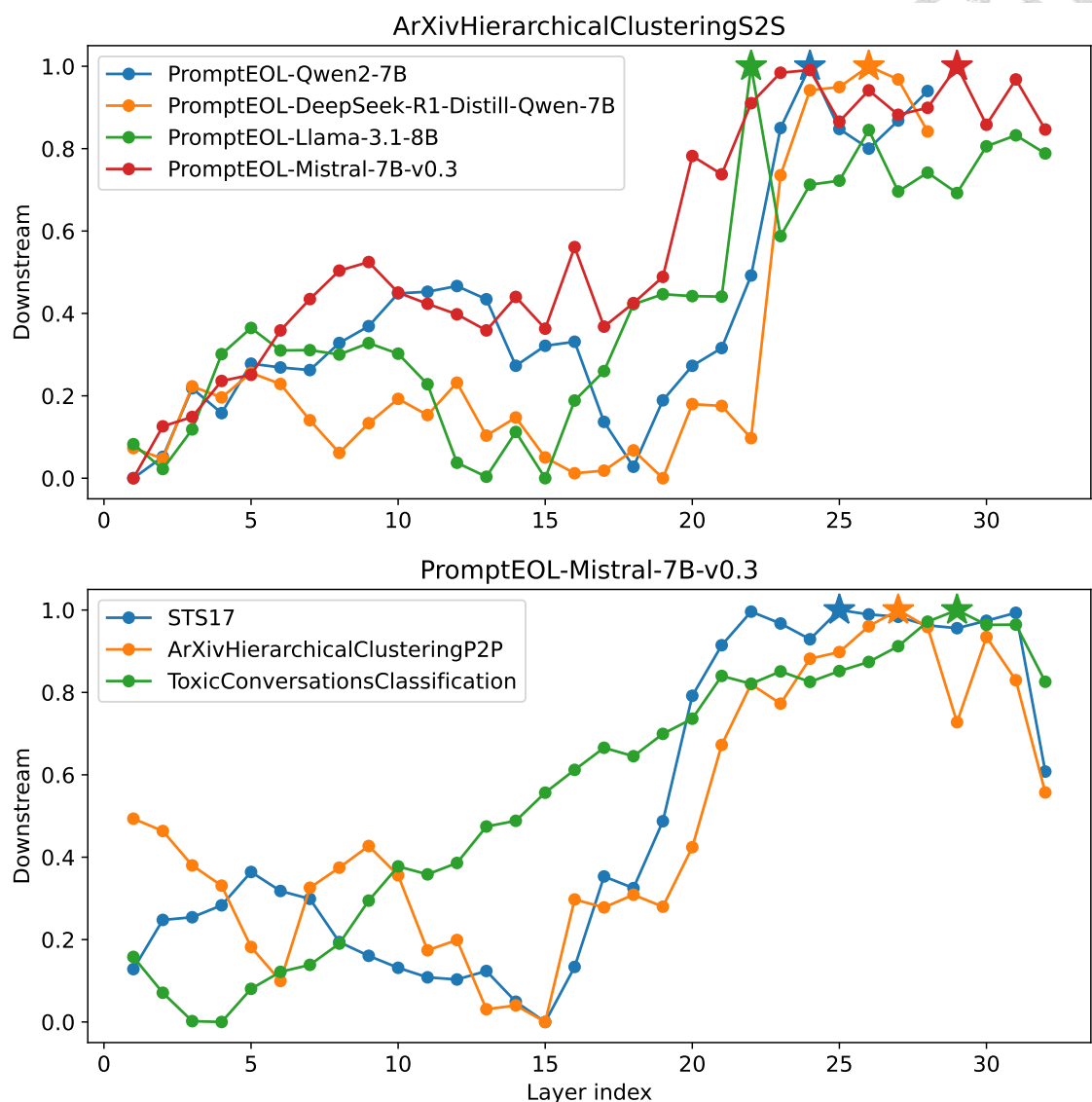
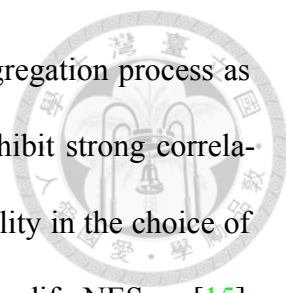


Figure 1.1: Last layer does not always yield best performance (the STAR sign stands for the best layer). For clearer visualization in the figure, the downstream performance of each line is normalized to a range between 0 and 1.

without supervision from the downstream tasks.

Building on these observations, we introduce LayerQW (**Layer-wise Quality Weighting**), a novel training-free framework designed to enhance text embeddings through quality-aware multi-layer pooling. Unlike conventional approaches that rely solely on the last layer representation, LayerQW adaptively integrates information from all layers of a large language model, enabling the extraction of richer semantic features and delivering more robust performance across a broad spectrum of natural language processing tasks. To de-



termine the optimal contribution of each layer, we formulate the aggregation process as an optimization problem guided by intrinsic quality metrics that exhibit strong correlations with downstream performance. This formulation allows flexibility in the choice of optimization algorithms. To effectively guide the optimization, we modify NESum [15], originally developed for image representations, into a text-compatible variant (NESum-Text) and employ it as a surrogate objective function. Our empirical analysis shows that NESum-Text exhibits a strong correlation with downstream performance on text embedding tasks, establishing it as a reliable proxy for model selection in scenarios without labeled data.

Extensive experiments conducted across a diverse set of NLP benchmarks validate the effectiveness of our approach. Compared to traditional strategies that utilize only the last token embedding from the last layer, LayerQW consistently achieves superior performance. These results underscore the importance of multi-layer integration and quality-aware pooling in producing robust and generalizable text representations. We summarize our contributions as follows:

- Through empirical study, we report that using only the last layer is not an ideal means to construct embeddings.
- We formulate the embedding generation task as an optimization task to linearly aggregate embeddings of all layers. Without supervised signal from the downstream performance, our optimization strategy relies on a carefully designed task-agnostic quality indicator based on the whitening degree of the representation.
- Extensive experiments were conducted, covering five training-free methods, four base models, and 42 downstream datasets across seven task categories. Results

consistently show that the proposed method yields significant improvement across a wide range of scenarios.





Chapter 2 Related Work

We categorize existing text embedding models based on two criteria: whether the model is LLM-based and whether it has undergone training. This classification results in four distinct categories, as summarized in Table 2.1. Below, we provide a more detailed introduction to the embedding models that are based on LLMs.

Table 2.1: Categorization of existing text embedding models by LLM dependency and training supervision.

Criteria	Non-LLM-Based	LLM-Based
Training-Based	SimCSE [13], DiffCSE [9], PromptBERT [19]	E5-Mistral [36], LLM2Vec [6], GritLM [26]
Training-Free	BERT [11], RoBERTa [24], ST5 [28]	Echo [31], PromptEOL [18], MetaEOL [22]

2.1 Training-Based Methods

These methods adopt contrastive learning to pull the query and its positive document closer in the embedding space, while pushing the query away from negative documents. E5-mistral-7b-instruct [36] utilizes a decoder-only LLM, enhanced through fine-tuning on diverse synthetic datasets generated by GPT-4, thereby eliminating the need for complex multi-stage training pipelines. LLM2Vec [6] introduces a method to repurpose generative LLMs into robust text encoders by applying bidirectional attention mechanisms, aligning them more closely with traditional encoder-based architectures. GritLM-7B [26] is trained to jointly optimize for both generative and embedding performance, bridging

the gap between language generation and representation learning within a single unified model. Bge-en-icl [23] further enhances embedding performance by leveraging few-shot in-context learning.



2.2 Training-Free Methods

Several works have proposed different prompts that enable LLMs to generate high-quality text embeddings without any training. Echo [31] repeats the input sentence to improve token-level attention and capture richer semantic information. PromptEOL [18] asks the LLM to represent a sentence using a single word (This sentence: '[TEXT]' means in one word:"). Then, the embedding of the " token is used as the sentence embedding. PCoTEOL [37] introduces a simulated chain-of-thought prompting strategy, while KEEOL [37] prepends a declarative background definition about sentence embeddings to guide the LLM's representation. MetaEOL [22] aggregates outputs from eight prompts spanning four meta-tasks to enhance robustness. GenEOL [34] employs LLMs as both embedder and generator in constructing embeddings, enabling a generative encoding process tailored for semantic understanding.



Chapter 3 Methodology

3.1 Problem Definition

Given the last token embedding from each hidden layer of a pretrained language model, denoted as $\mathbf{E} = \{e_1, e_2, \dots, e_n\}$, our goal is to learn a set of weights $\mathbf{w} = \{w_1, w_2, \dots, w_n\}$ to aggregate these layer-wise embeddings into a single final representation:

$$\mathbf{E}_{final} = \mathbf{w}\mathbf{E} = \sum_{i=1}^n w_i e_i \quad (3.1)$$

where $\mathbf{E}_{final}$ denotes the final embedding. Ideally, the best weights $\mathbf{w}^*$ should be optimized to maximize performance on downstream tasks $\mathbf{D}$:

$$\mathbf{w}^* = \arg \max_{\mathbf{w}} \mathbf{D}(\mathbf{E}_{final}) \quad (3.2)$$

However, since the specific downstream tasks are unknown, we propose to optimize the weights based on a surrogate quality metric $\mathbf{Q}$ that correlates with downstream performance:

$$\mathbf{w}^* = \arg \max_{\mathbf{w}} \mathbf{Q}(\mathbf{E}_{final}) \quad (3.3)$$



This predefined metric allows us to estimate the utility of each layer's representation without requiring supervision from downstream tasks. The final embedding is then computed as the weighted sum of the last token embeddings across all layers, guided by these learned weights.

3.2 Quality Metric Design for Text Embeddings

Inspired by NESum [15] from the image domain, we design NESum-Text, a variant adapted for textual data. Given a text dataset $\mathbf{T}$ of length N , we first apply a text embedding model ϕ to obtain an embedding matrix $\mathbf{M} = \phi(\mathbf{T}) \in \mathbb{R}^{N \times d}$, where each row is a d -dimensional embedding vector and N is the number of samples. To capture the linear relationships among embedding dimensions, we compute the covariance matrix $\mathbf{C} \in \mathbb{R}^{d \times d}$, defined as:

$$\mathbf{C} = \frac{1}{N-1} (\mathbf{M} - \bar{\mathbf{M}})^\top (\mathbf{M} - \bar{\mathbf{M}}) \quad (3.4)$$

Here, let $\mathbf{m}_i$ denote the i -th row of $\mathbf{M}$. Then, $\bar{\mathbf{M}}$ is a matrix where each row is the mean vector $\bar{\mathbf{m}} = \frac{1}{N} \sum_{i=1}^N \mathbf{m}_i$. Since the covariance matrix $\mathbf{C}$ is symmetric and positive semi-definite, we can perform eigendecomposition:

$$\mathbf{C} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top \quad (3.5)$$

where $\mathbf{V} \in \mathbb{R}^{d \times d}$ is an orthogonal matrix whose columns are the eigenvectors of $\mathbf{C}$, and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_d)$ is a diagonal matrix containing the non-negative eigenvalues of $\mathbf{C}$. NESum-Text (Normalized Eigenvalue Sum) is then calculated as follows:

$$\text{NESum-Text}(\mathbf{\Lambda}) = \sum_{i=1}^d \frac{\lambda_i}{\max_j \lambda_j} \quad (3.6)$$

This metric is used to quantify the degree of whitening in an embedding. That is, it measures how evenly the features are distributed across different dimensions. In the context of text embeddings, whitening has been shown to enhance representation quality by removing dominant directions and encouraging isotropic distributions [12, 32]. When only a few eigenvalues of the covariance matrix dominate, the NESum-Text score becomes low, indicating that the representation is concentrated in limited directions and lacks balance. Conversely, when the eigenvalues are more uniform in magnitude, the NESum-Text score increases, reflecting a more isotropic and well-whitened embedding space. A higher NESum-Text thus suggests reduced risk of the curse of dimensionality, leading to better generalization and improved downstream task performance.

3.3 Bayesian Optimization with TPE

Since NESum-Text is non-differentiable, traditional gradient-based optimization methods cannot be directly applied. To address this challenge, we adopt Bayesian optimization, a technique widely used for solving black-box optimization problems [30].

In Bayesian Optimization, the goal is to maximize the objective function $y = f(x)$ as follows:



$$x^* \in \arg \max_{x \in \mathcal{X}} f(x) \quad (3.7)$$

where x denotes a hyperparameter configuration, x^* represents the optimal configuration, and $\mathcal{X}$ is the search space containing all valid configurations. Bayesian Optimization iteratively searches for the optimal x^* using an acquisition function, which balances exploration and exploitation.

Specifically, we utilize the Tree-structured Parzen Estimator (TPE) algorithm [7], as implemented in the Optuna framework [5]. TPE models the objective function by dividing the search space into two densities with respect to a predefined threshold y^γ :

$$p(x | y) = \begin{cases} \ell(x) & \text{if } y \geq y^\gamma \\ g(x) & \text{if } y < y^\gamma \end{cases} \quad (3.8)$$

Here, $p(x | y)$ is the conditional probability density, where $\ell(x)$ represents the density over better configurations (i.e., those with $y \geq y^\gamma$), and $g(x)$ over worse ones. Both $\ell(x)$ and $g(x)$ are modeled using Parzen estimators, which are kernel density estimators (KDEs) approximating distributions from observed samples.

The optimization of the expected improvement in TPE is equivalent to maximizing the ratio $\ell(x)/g(x)$. The configuration selected at iteration t is therefore given by:

$$x_t = \arg \max_{x \in \mathcal{X}} \frac{\ell(x)}{g(x)} \quad (3.9)$$

By leveraging kernel-based density estimation and a flexible non-parametric model-

ing strategy, TPE can efficiently navigate high-dimensional and irregular search spaces, making it particularly suitable for optimizing our non-differentiable quality metric.





Chapter 4 Experiment

4.1 Experiment Setup

4.1.1 Datasets

To compute the quality metric scores, we randomly sampled 10,000 sentences from English Wikipedia. Given the task-agnostic nature of the quality metrics, we intentionally refrained from using any labeled datasets for this evaluation. For assessing downstream performance, we utilized the Massive Text Embedding Benchmark (MTEB) [27] to evaluate downstream scores across a variety of datasets. To demonstrate the general applicability of our method, we conducted tests on 42 datasets, encompassing seven distinct task categories: Semantic Textual Similarity (STS), Classification, Pair Classification, Clustering, Retrieval, Ranking, and Summarization. The statistics for all datasets are detailed in Table 4.1.

4.1.2 Training-Free Methods

We evaluate LayerQW using five training-free embedding strategies: Echo, PromptEOL, PCoTEOL, KEEOL, and MetaEOL. Prompt-based methods were originally proposed to enhance sentence embeddings without additional training. This experimental setup en-

Table 4.1: Statistics of datasets.

Category	Evaluation Metric	Datasets
STS	Spearman Correlation	BIOSSES, SICK-R, STS12, STS13, STS14, STS15, STS16, STS17, STS22, STSBenchmark
Retrieval	NDCG@10	ArguAna, NFCorpus, SCIDOCS, SciFact
Classification	Accuracy	AmazonCounterfactualClassification, AmazonReviewsClassification, Banking77Classification, EmotionClassification, MassiveIntentClassification, MassiveScenarioClassification, MTOPDomainClassification, MTOPIntentClassification, ToxicConversationsClassification, TweetSentimentExtractionClassification
PairClassification	Maximum Average Precision	SprintDuplicateQuestions, TwitterSemEval2015, TwitterURLCorpus
Clustering	V-Measure	ArXivHierarchicalClusteringP2P, ArXivHierarchicalClusteringS2S, BiorxivClusteringP2P, BiorxivClusteringS2S, MedrxivClusteringP2P, MedrxivClusteringS2S, RedditClustering, StackExchangeClusteringP2P, StackExchangeClustering, TwentyNewsgroupsClustering
Reranking	Mean Average Precision	AskUbuntuDupQuestions, MindSmallReranking, SciDoc-sRR, StackOverflowDupQuestions
Summarization	Spearman Correlation	SummEvalSummarization

ables a comprehensive assessment of the generalizability of LayerQW across a diverse range of prompting paradigms. For each training-free method, we extract the last token embedding from the large language model to serve as the text representation. This consistent embedding strategy ensures a fair comparison across all evaluated methods.

4.1.3 Base Models

To assess the generalizability of our approach across diverse model architectures and training paradigms, we adopt four large language models: DeepSeek-R1-Distill-Qwen-7B [10], Mistral-7B [17], Qwen2-7B [2], and LLaMA3.1-8B [35]. All selected models are publicly available via the Hugging Face Model Hub and represent a range of open-weight LLMs with varying design choices and pretraining objectives.



4.1.4 Baselines

In addition to using the last layer representation (Last) as a standard baseline, we implement a uniform layer averaging strategy (Avg-All), where equal weights are assigned to all layers during aggregation. To further establish a strong baseline, we also consider averaging the last 10 layers of the model (Avg-Last10), motivated by prior observations that the upper layers typically capture more task-relevant information.

4.1.5 Statistical Test

Following prior work [33], we apply the Wilcoxon Signed-Rank Test to assess whether the performance gains of our method over baseline approaches are statistically significant. This non-parametric test is well-suited for comparing paired results across multiple datasets without assuming normality. We consider results with p-values less than 0.05 to indicate statistically significant differences.

4.2 Main Results

Table 4.2 and Table 4.3 present a comprehensive evaluation of various embedding combination strategies across five training-free prompting methods and four base models. Each column in the table represents the average downstream score across all datasets within the corresponding task type. The last column, Avg., indicates the average score across all 42 datasets. To facilitate interpretation, we mark LayerQW results with asterisks (*) to indicate statistically significant improvements over the strongest baseline (i.e., the best of Last, Avg-All, or Avg-Last10). In addition, the values in parentheses indicate

Table 4.2: Experimental results on DeepSeek-R1-Distill-Qwen-7B and Mistral-7B.

Prompt	Method	STS	PairClass.	Clust.	Rerank	Class.	Retr.	Summ.	Avg.
<i>Base Model: DeepSeek-R1-Distill-Qwen-7B</i>									
Echo	Last	23.74	31.25	25.06	33.45	46.72	0.85	1.50	28.28
	Avg-All	21.33	30.78	24.32	33.18	46.62	0.81	1.62	27.44
	Avg-Last10	20.85	30.50	24.20	33.19	45.51	0.79	6.64	27.13
	LayerQW	24.18	32.10	24.23	34.13	50.51	0.80	9.16	29.39 (+3.95%)*
PromptEOL	Last	53.76	62.69	36.63	47.68	66.29	8.62	8.47	47.35
	Avg-All	53.80	55.22	35.25	46.32	66.77	6.44	13.78	46.40
	Avg-Last10	55.44	56.04	36.60	47.31	67.03	8.47	16.65	47.59
	LayerQW	58.13	57.97	38.34	48.49	67.88	12.95	18.59	49.57 (+4.16%)*
PCoTEOL	Last	54.41	61.81	35.63	47.20	66.25	7.28	11.51	47.09
	Avg-All	54.11	54.70	35.28	46.59	67.27	5.59	14.35	46.52
	Avg-Last10	55.52	55.24	36.21	47.30	67.31	7.74	16.56	47.45
	LayerQW	57.22	57.03	37.54	48.25	68.03	10.90	18.65	48.91 (+3.08%)*
KEEOL	Last	51.77	66.03	35.45	46.32	67.31	6.99	11.92	46.87
	Avg-All	50.61	55.90	34.14	46.20	67.63	3.62	7.07	45.19
	Avg-Last10	52.05	56.94	35.20	46.72	67.96	4.19	10.31	46.12
	LayerQW	54.37	59.67	36.88	47.92	68.74	5.83	12.45	47.77 (+1.93%)*
MetaEOL	Last	54.79	69.21	35.23	46.69	69.54	17.70	15.30	49.43
	Avg-All	57.67	62.28	34.82	47.23	69.94	5.56	12.03	48.43
	Avg-Last10	59.81	63.90	36.49	47.93	70.23	7.77	16.08	49.90
	LayerQW	60.17	65.15	36.82	48.35	70.60	8.89	16.15	50.39 (+0.98%)*
<i>Base Model: Mistral-7B</i>									
Echo	Last	38.19	42.16	29.44	42.66	59.13	1.87	4.79	37.55
	Avg-All	34.77	38.41	28.85	41.02	57.76	1.51	4.67	35.81
	Avg-Last10	36.05	39.57	29.08	41.61	59.06	1.56	5.00	36.63
	LayerQW	39.76	44.36	29.83	43.60	59.83	2.19	5.55	38.47 (+2.47%)*
PromptEOL	Last	63.43	62.07	38.88	51.23	69.11	22.60	24.20	52.86
	Avg-All	66.68	62.26	39.70	52.31	70.20	23.38	26.91	54.34
	Avg-Last10	66.54	62.62	39.67	52.43	70.07	23.89	26.52	54.34
	LayerQW	67.67	62.25	40.04	52.48	70.80	23.66	26.88	54.84 (+0.92%)*
PCoTEOL	Last	62.46	64.53	37.51	52.18	68.85	21.14	22.20	52.31
	Avg-All	66.31	64.81	38.96	53.33	70.06	23.52	24.37	54.27
	Avg-Last10	66.21	65.02	38.84	53.48	69.84	23.90	24.38	54.24
	LayerQW	67.53	64.41	39.80	53.56	70.81	25.11	27.48	55.16 (+1.64%)*
KEEOL	Last	65.26	70.73	35.62	49.11	70.56	22.57	22.34	53.23
	Avg-All	67.13	69.74	36.90	50.25	71.62	23.40	25.27	54.42
	Avg-Last10	67.27	70.20	36.74	50.37	71.46	24.17	25.96	54.51
	LayerQW	67.23	68.78	37.59	50.77	72.22	23.34	26.42	54.75 (+0.45%)*
MetaEOL	Last	69.05	77.63	40.93	53.10	73.00	30.44	23.93	57.64
	Avg-All	70.89	77.13	41.83	53.58	73.87	30.31	25.72	58.54
	Avg-Last10	70.75	77.31	41.87	53.63	73.72	31.10	26.08	58.58
	LayerQW	70.85	77.23	42.08	53.55	73.92	30.77	25.86	58.65 (+0.13%)*

the percentage improvement of LayerQW relative to the best-performing baseline, providing a more intuitive understanding of the relative gain.

The experimental results clearly demonstrate that LayerQW consistently outperforms both naive averaging across layers and the use of the original last layer representation.

Table 4.3: Experimental results on Qwen2-7B and LLaMA3.1-8B.

Prompt	Method	STS	PairClass.	Clust.	Rerank	Class.	Retr.	Summ.	Avg.
<i>Base Model: Qwen2-7B</i>									
Echo	Last	28.97	38.71	26.21	37.15	53.92	0.87	-2.07	32.31
	Avg-All	24.21	32.32	25.70	34.99	51.79	0.84	-4.54	29.83
	Avg-Last10	24.58	32.72	25.63	35.43	51.30	0.77	-3.36	29.87
	LayerQW	29.54	38.81	25.63	37.79	56.04	1.06	-2.87	32.88 (+1.77%)
PromptEOL	Last	62.51	69.81	40.49	51.79	70.50	22.92	12.09	53.70
	Avg-All	63.09	62.12	40.61	51.46	72.17	22.82	22.37	53.92
	Avg-Last10	63.73	62.77	40.99	51.97	72.16	26.77	26.83	54.73
	LayerQW	65.19	64.79	41.74	52.63	72.55	28.90	25.40	55.73 (+1.83%)*
PCoTEOL	Last	62.87	68.57	39.41	51.55	71.00	20.44	14.60	53.36
	Avg-All	62.59	61.30	38.61	50.61	72.40	15.19	23.87	52.54
	Avg-Last10	63.16	61.84	39.03	50.99	72.30	19.33	27.18	53.31
	LayerQW	64.62	64.43	40.03	52.19	72.85	23.64	24.85	54.68 (+2.47%)*
KEEOL	Last	64.63	71.57	37.63	49.50	70.96	20.50	12.77	53.32
	Avg-All	63.88	62.56	37.05	47.93	72.04	9.72	20.21	51.63
	Avg-Last10	64.60	63.15	37.12	48.27	72.02	14.63	25.21	52.47
	LayerQW	66.70	67.95	37.82	50.45	72.35	24.38	28.11	54.76 (+2.69%)*
MetaEOL	Last	64.16	76.28	40.23	50.70	72.69	26.54	21.64	55.48
	Avg-All	65.95	66.57	39.02	49.55	73.60	11.67	23.80	53.67
	Avg-Last10	66.78	67.13	40.02	49.89	73.59	19.86	29.31	55.08
	LayerQW	67.55	70.03	40.34	50.87	74.00	22.20	27.27	55.92 (+0.79%)
<i>Base Model: Llama3.1-8B</i>									
Echo	Last	39.42	42.27	31.25	42.18	58.58	2.04	-4.15	37.90
	Avg-All	31.76	35.73	28.53	38.83	53.31	0.99	-5.69	33.26
	Avg-Last10	35.26	38.32	29.09	40.42	56.91	1.17	-5.78	35.43
	LayerQW	41.43	43.07	30.49	43.31	59.27	2.10	-3.57	38.55 (+1.71%)
PromptEOL	Last	65.47	60.43	40.71	51.92	69.24	26.58	25.97	54.18
	Avg-All	65.91	57.97	38.50	50.81	71.01	26.47	25.44	53.88
	Avg-Last10	66.88	59.23	39.31	51.67	70.97	29.79	26.32	54.80
	LayerQW	67.13	59.66	39.63	51.88	71.07	29.77	26.69	55.02 (+0.40%)*
PCoTEOL	Last	67.66	62.55	41.29	53.27	69.06	27.36	23.09	55.08
	Avg-All	68.07	60.61	40.13	52.39	70.82	28.31	24.50	55.22
	Avg-Last10	68.80	61.62	40.59	53.10	70.81	31.16	25.74	55.95
	LayerQW	69.05	62.20	41.07	53.36	70.88	31.25	26.07	56.22 (+0.48%)*
KEEOL	Last	67.79	66.86	41.34	52.42	70.47	23.96	22.35	55.34
	Avg-All	66.40	63.27	38.82	51.21	72.12	24.49	22.52	54.49
	Avg-Last10	67.55	64.85	40.02	52.11	72.12	27.57	23.89	55.57
	LayerQW	67.68	69.21	40.21	52.27	71.88	29.98	22.20	56.11 (+0.96%)
MetaEOL	Last	71.37	77.22	44.67	53.73	72.23	29.04	18.11	58.66
	Avg-All	71.18	74.03	43.00	53.47	74.12	28.38	22.32	58.45
	Avg-Last10	72.01	75.61	44.11	54.01	74.05	33.36	23.92	59.57
	LayerQW	71.94	75.79	43.75	53.89	74.30	31.47	22.96	59.33 (-0.41%)

This trend is especially pronounced under the PromptEOL prompting strategy with the DeepSeek-R1-Distill-Qwen-7B model, where LayerQW achieves an average performance improvement of 4.16% over the strongest baseline, with the difference being statistically significant. Similar patterns are observed across other prompting methods and base models, suggesting that LayerQW generalizes well across different configurations and task

settings.

Our analysis confirms that multi-layer pooling often yields more informative representations than using only the last layer. Even simple average pooling across all layers improves performance in many cases. This improvement is particularly notable for Mistral-7B, where PromptEOL and PCoTEOL improve from 52.86 to 54.34 and from 52.31 to 54.27, respectively. These gains suggest that earlier layers capture complementary semantic information. However, we also observe instances where Avg-All degrades performance, notably in the Echo prompting condition. This indicates that naive averaging may suppress important task-specific signals.

We compare the performance of Avg-All and Avg-Last10. Our results show that Avg-Last10 consistently outperforms Avg-All across most settings, with only two exceptions: DeepSeek-R1-Distill-Qwen-7B with Echo and Mistral-7B with PCoTEOL, where Avg-Last10 yields slightly lower performance. This aligns with prior observations that the final layers of large language models tend to capture more task-relevant information. As a result, aggregating only the last 10 layers generally leads to better downstream performance.

Crucially, LayerQW consistently achieves superior performance regardless of whether uniform averaging improves or degrades downstream task outcomes. For instance, under the PromptEOL setting with the Qwen2-7B backbone, Avg-All improves the score from 53.70 to 53.92, and Avg-Last10 further raises it to 54.73. LayerQW surpasses both baselines, reaching 55.73 with a statistically significant improvement of 1.83%. In addition, under the Echo setting with the Mistral-7B backbone, Avg-All leads to a performance drop from 37.55 to 35.81. Although Avg-Last10 partially recovers the performance to 36.63, it still lags behind Last. In contrast, LayerQW reverses this decline and boosts performance

to 38.47, yielding a statistically significant gain of 2.47%.

In summary, LayerQW emerges as a robust and versatile approach for aggregating multi-layer representations in large language models. It consistently yields improved downstream performance over conventional methods, maintains effectiveness across diverse prompts and architectures, and achieves statistically significant gains in nearly all evaluated cases. These findings underscore LayerQW potential as a principled and reliable method for enhancing sentence representation quality in training-free settings.

4.3 The Upper Bound of Task-Agnostic Optimization

To estimate the upper bound of our method, we conduct an oracle experiment where we assume access to downstream task performance during the optimization process. Specifically, we define the objective function as the average performance across all 10 STS datasets, thereby allowing the optimizer to directly optimize for true downstream performance rather than relying on a proxy metric. This setting simulates an ideal case where downstream supervision is available during search, which is not a feasible scenario in real-world deployment but useful for understanding the ceiling of task-agnostic methods. The experiment is conducted using the LLaMA3.1-8B backbone and PromptEOL configuration.

The results are summarized in Table 4.4. We denote this ideal result as Oracle, and for each method, we also report the relative performance gap to Oracle in parentheses. Notably, LayerQW achieves performance remarkably close to this upper bound, with only a 0.28% relative difference. This small gap highlights the effectiveness of our optimization framework in the absence of downstream labels. The closeness to Oracle indicates that

our task-agnostic quality metric (NESum-Text) successfully aligns with actual task performance, despite not relying on any supervision. This suggests that NESum-Text captures intrinsic embedding properties that are robust and transferable across diverse downstream tasks.

Table 4.4: Upper bound estimation using downstream-aware optimization.

Method	BIOSSES	SICK-R	STS12	STS13	STS14	STS15	STS16	STS17	STS22	STSB.	STS Avg.
Last	74.10	70.44	60.77	80.15	69.64	76.89	78.23	59.88	11.98	72.65	65.47 (-2.75%)
Avg-All	72.61	71.02	60.14	80.61	69.93	77.26	77.26	66.39	10.61	73.30	65.91 (-2.09%)
Avg-Last10	73.63	71.66	61.67	81.82	71.27	78.28	78.31	67.57	9.95	74.63	66.88 (-0.65%)
LayerQW	74.49	71.66	61.80	81.70	71.19	78.29	78.48	67.67	11.45	74.59	67.13 (-0.28%)
Oracle	74.79	71.66	61.93	81.64	71.18	78.38	78.72	67.59	12.84	74.50	67.32



Chapter 5 Ablation Study

5.1 Comparison of Different Objective Function

In addition to NESum, other quality metrics such as α -ReQ and RankMe can also be used to assess the quality of text embeddings. α -ReQ [4] captures the spectral decay behavior of representations by modeling the distribution of eigenvalues with a power-law. RankMe [14] evaluates the effective dimensionality of a representation by computing the entropy of its normalized singular values. We compare the performance of different quality metrics when used as the objective function in our framework. Specifically, we fix the training-free method to PromptEOL and conduct this experiment across four base models: LLaMA3.1-8B, Mistral-7B, DeepSeek-R1-Distill-Qwen-7B, and Qwen2-7B.

The results, shown in Table 5.1, report downstream performance across seven task types, as well as the average score across all tasks. We observe that embeddings optimized using NESum consistently achieve the best overall performance on three out of the four models: LLaMA3.1-8B, Mistral-7B, and DeepSeek-R1-Distill-Qwen-7B. In the case of Qwen2-7B, RankMe slightly outperforms NESum by a narrow margin in the average score (55.77 vs. 55.73), although NESum still performs best on most individual tasks. This further supports the use of NESum as a reliable and effective quality metric for guiding training-free model selection.

Table 5.1: Downstream performance optimized by various quality metrics.

Model	Quality	STS	PairClass.	Clust.	Rerank	Class.	Retr.	Summ.	Avg.
Llama3.1-8B	NESum	67.13	59.66	39.63	51.88	71.07	29.77	26.69	55.02
	RankMe	66.57	58.98	39.07	51.35	71.09	29.38	26.09	54.60
	α -ReQ	63.83	55.58	37.05	49.08	70.64	21.96	22.63	52.11
Mistral-7B	NESum	67.67	62.25	40.04	52.48	70.80	23.66	26.88	54.84
	RankMe	67.26	61.80	40.01	52.13	70.83	22.88	27.10	54.61
	α -ReQ	65.94	60.43	39.24	51.34	70.58	20.42	25.78	53.61
DeepSeek-R1-Distill-Qwen-7B	NESum	58.13	57.97	38.34	48.49	67.88	12.95	18.59	49.57
	RankMe	57.29	57.50	37.58	48.14	67.70	10.57	18.08	48.83
	α -ReQ	54.96	56.96	34.81	47.01	67.55	6.99	15.05	47.03
Qwen2-7B	NESum	65.19	64.79	41.74	52.63	72.55	28.90	25.40	55.73
	RankMe	65.12	64.38	41.96	52.60	72.55	29.16	25.75	55.77
	α -ReQ	61.85	62.23	38.99	50.96	72.20	15.96	21.81	52.54

To further validate the effectiveness of each quality metric, we compute their correlation with actual downstream performance across datasets. Specifically, we calculate quality scores for each layer of the four base models using PromptEOL. For each model, we then evaluate the Pearson correlation between the layer-wise quality scores and the corresponding downstream task performance. The results, presented in Table 5.2, show that NESum consistently achieves the highest correlation, followed by RankMe and α -ReQ. These findings suggest that NESum aligns more closely with real-world task performance and serves as a more reliable proxy objective.

Table 5.2: Correlations between quality metric scores and downstream performance.

Quality	STS	PairClass.	Clust.	Rerank	Class.	Retr.	Summ.	Avg.
NESum	83.05	77.27	82.96	89.15	76.05	94.31	86.82	82.91
RankMe	86.95	78.42	72.64	87.75	83.46	82.00	88.32	81.89
α -ReQ	64.02	63.31	49.96	67.96	76.17	57.60	64.63	63.31

5.2 Comparison of Different Samplers

We compare the performance of our selected sampling strategy, Tree-structured Parzen Estimator (TPE), with a commonly used heuristic method, the Genetic Algorithm (GA) [1].

The comparison is conducted under a training-free setting using PromptEOL, where LLaMA3.1-

8B serves as the base model for generating embeddings. As presented in Table 5.3, TPE achieves slightly better average performance than GA across the evaluated tasks (55.02 vs. 54.95). However, statistical significance testing reveals that the performance difference is not significant, indicating that both methods are comparably effective in our setting.

Table 5.3: Comparative results of different sampling strategies.

Sampler	STS	PairClass.	Clust.	Rerank	Class.	Retr.	Summ.	Avg.
TPE	67.13	59.66	39.63	51.88	71.07	29.77	26.69	55.02
GA	67.05	59.72	39.52	51.82	71.07	29.45	26.97	54.95

Overall, our findings demonstrate that the proposed framework is robust to the choice of sampling method. While TPE offers better efficiency and slightly more stable results, the compatibility with alternative strategies like GA highlights the general applicability and adaptability of our approach in black-box optimization scenarios.

5.3 Comparison of Different Trial Numbers

The number of optimization trials is treated as a tunable hyperparameter in our framework. We adopt PromptEOL as the training-free method and use Mistral-7B as the base model to investigate how varying the number of trials affects embedding quality and downstream performance.

As shown in Table 5.4, increasing the number of trials from 10 to 100 leads to consistent improvements across most task types. However, performance plateaus beyond 100 trials, with average scores remaining stable and even slight drops observed in some tasks. This suggests limited additional value from further exploration.

Each trial only requires computing the NESum-Text score, which is lightweight and

training-free. Therefore, running hundreds of trials is still efficient. Nonetheless, 100 trials strike a good balance between performance and efficiency, making it a practical default.

Table 5.4: Performance comparison across different numbers of optimization trials.

Trials	STS	PairClass.	Clust.	Rerank	Class.	Retr.	Summ.	Avg.
10	66.76	61.46	39.48	52.02	70.87	22.41	26.70	54.29
50	66.89	61.75	39.78	52.06	70.92	22.72	26.29	54.44
100	67.67	62.25	40.04	52.48	70.80	23.66	26.88	54.84
200	67.48	62.17	40.12	52.38	70.85	23.48	26.91	54.80
300	67.39	62.16	39.97	52.33	70.88	23.49	26.79	54.74



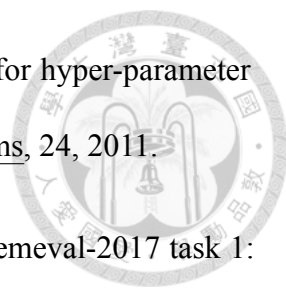
Chapter 6 Conclusion

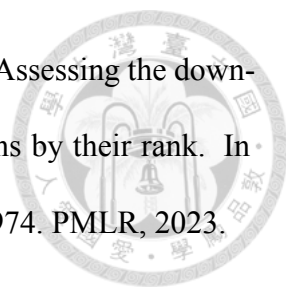
This work highlights the limitations of relying solely on last layer representations in large language models for text embedding, and demonstrates the benefits of aggregating information across all layers. By leveraging a task-agnostic quality metric originally developed for the image domain, we utilize a training-free, quality-aware framework that optimizes layer-wise integration without requiring labeled data. Extensive experiments across diverse models, tasks, and datasets confirm the robustness and generalizability of our approach. These findings underscore the importance of revisiting layer utilization strategies in LLMs and open new directions for embedding optimization in zero-shot and low-resource NLP settings.



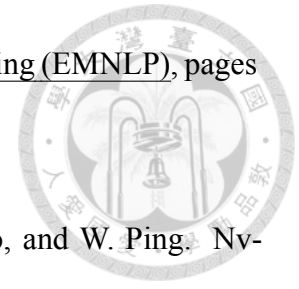
References

- [1] A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation, 6(2):182–197, 2002.
- [2] Qwen2 technical report. 2024.
- [3] E. Agirre, D. Cer, M. Diab, and A. Gonzalez-Agirre. Semeval-2012 task 6: A pilot on semantic textual similarity.* sem 2012: The first joint conference on lexical and computational semantics—. In Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012), Montréal, QC, Canada, pages 7–8, 2012.
- [4] K. K. Agrawal, A. K. Mondal, A. Ghosh, and B. Richards. α -req: Assessing representation quality in self-supervised learning by measuring eigenspectrum decay. Advances in Neural Information Processing Systems, 35:17626–17638, 2022.
- [5] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 2623–2631, 2019.
- [6] P. BehnamGhader, V. Adlakha, M. Mosbach, D. Bahdanau, N. Chapados, and S. Reddy. Llm2vec: Large language models are secretly powerful text encoders. arXiv preprint arXiv:2404.05961, 2024.

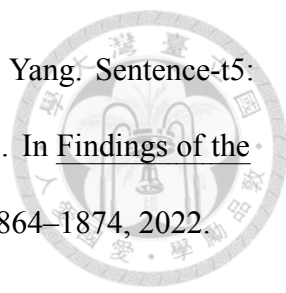
- 
- [7] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. Advances in neural information processing systems, 24, 2011.
- [8] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. arXiv preprint arXiv:1708.00055, 2017.
- [9] Y.-S. Chuang, R. Dangovski, H. Luo, Y. Zhang, S. Chang, M. Soljačić, S.-W. Li, S. Yih, Y. Kim, and J. Glass. Diffcse: Difference-based contrastive learning for sentence embeddings. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4207–4218, 2022.
- [10] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.
- [11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), pages 4171–4186, 2019.
- [12] S. Gao, S. Dou, Q. Zhang, and X.-J. Huang. Kernel-whitening: Overcome dataset bias with isotropic sentence embedding. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 4112–4122, 2022.
- [13] T. Gao, X. Yao, and D. Chen. Simcse: Simple contrastive learning of sentence embeddings. In EMNLP 2021-2021 Conference on Empirical Methods in Natural Language Processing, Proceedings, 2021.

- 
- [14] Q. Garrido, R. Balestrieri, L. Najman, and Y. Lecun. Rankme: Assessing the downstream performance of pretrained self-supervised representations by their rank. In International conference on machine learning, pages 10929–10974. PMLR, 2023.
- [15] B. He and M. Ozay. Exploring the gap between collapsed & whitened features in self-supervised learning. In International Conference on Machine Learning, pages 8613–8634. PMLR, 2022.
- [16] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave. Unsupervised dense information retrieval with contrastive learning. arXiv preprint arXiv:2112.09118, 2021.
- [17] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7b, 2023.
- [18] T. Jiang, S. Huang, Z. Luan, D. Wang, and F. Zhuang. Scaling sentence embeddings with large language models. In Findings of the Association for Computational Linguistics: EMNLP 2024, pages 3182–3196, 2024.
- [19] T. Jiang, J. Jiao, S. Huang, Z. Zhang, D. Wang, F. Zhuang, F. Wei, H. Huang, D. Deng, and Q. Zhang. Promptbert: Improving bert sentence embeddings with prompts. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 8826–8837, 2022.
- [20] P. Ke, H. Ji, S. Liu, X. Zhu, and M. Huang. Sentilare: Sentiment-aware language representation learning with linguistic knowledge. In Proceedings of the 2020

Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6975–6988, 2020.



- [21] C. Lee, R. Roy, M. Xu, J. Raiman, M. Shoeybi, B. Catanzaro, and W. Ping. Nv-embed: Improved techniques for training llms as generalist embedding models. arXiv preprint arXiv:2405.17428, 2024.
- [22] Y. Lei, D. Wu, T. Zhou, T. Shen, Y. Cao, C. Tao, and A. Yates. Meta-task prompting elicits embeddings from large language models. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 10141–10157, 2024.
- [23] C. Li, M. Qin, S. Xiao, J. Chen, K. Luo, Y. Shao, D. Lian, and Z. Liu. Making text embedders few-shot learners. arXiv preprint arXiv:2409.15700, 2024.
- [24] Z. Liu, W. Lin, Y. Shi, and J. Zhao. A robustly optimized bert pre-training approach with post-training. In China national conference on Chinese computational linguistics, pages 471–484. Springer, 2021.
- [25] B. Mitra, F. Diaz, and N. Craswell. Learning to match using local and distributed representations of text for web search. In Proceedings of the 26th international conference on world wide web, pages 1291–1299, 2017.
- [26] N. Muennighoff, S. Hongjin, L. Wang, N. Yang, F. Wei, T. Yu, A. Singh, and D. Kiela. Generative representational instruction tuning. In ICLR 2024 Workshop: How Far Are We From AGI.
- [27] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers. Mteb: Massive text embedding benchmark. In Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 2014–2037, 2023.

- 
- [28] J. Ni, G. H. Abrego, N. Constant, J. Ma, K. Hall, D. Cer, and Y. Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In Findings of the Association for Computational Linguistics: ACL 2022, pages 1864–1874, 2022.
- [29] O. Skean, M. R. Arefin, D. Zhao, N. Patel, J. Naghiyev, Y. LeCun, and R. Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. arXiv preprint arXiv:2502.02013, 2025.
- [30] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems, 25, 2012.
- [31] J. M. Springer, S. Kotha, D. Fried, G. Neubig, and A. Raghunathan. Repetition improves language model embeddings. arXiv preprint arXiv:2402.15449, 2024.
- [32] J. Su, J. Cao, W. Liu, and Y. Ou. Whitening sentence representations for better semantics and faster retrieval. arXiv preprint arXiv:2103.15316, 2021.
- [33] Y. Tang and Y. Yang. Pooling and attention: What are effective designs for llm-based embedding models? arXiv preprint arXiv:2409.02727, 2024.
- [34] R. Thirukovalluru and B. Dhingra. Geneol: Harnessing the generative power of llms for training-free sentence embeddings. arXiv preprint arXiv:2410.14635, 2024.
- [35] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023.
- [36] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei. Improving text embeddings with large language models. In Proceedings of the 62nd Annual Meeting

of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11897–11916, 2024.

- [37] B. Zhang, K. Chang, and C. Li. Simple techniques for enhancing sentence embeddings in generative language models. In International Conference on Intelligent Computing, pages 52–64. Springer, 2024.