# Beyond Single Representations: Multi-Model Embedding Fusion for Stable Text Classification

**Jiho Gwak, Yuchul Jung***
Kumoh National Institute of Technology, South Korea
{wlgh6709, jyc}@kumoh.ac.kr

## Abstract

Embedding fusion has become a widely adopted technique for enhancing performance across various NLP tasks. While prior research suggests that different layers of language models encode distinct representations and that pooling strategies influence performance, there is a lack of systematic analysis regarding the empirical efficacy of these differences or the impact of combining embeddings from multiple models. This study provides a rigorous, empirical evaluation of layer-wise fusion strategies to determine their actual contribution to classification performance. Our findings reveal that the effectiveness of individual layers is more dependent on dataset characteristics than on the model architecture itself. Furthermore, we demonstrate that fusing embeddings from multiple models yields more robust and consistent representations across tasks, with the influence of any single model diminishing as the number of integrated models increases. Notably, experiments on low-resource datasets show that embedding fusion provides particularly significant gains when training data is scarce, highlighting its robustness and adaptability in data-constrained environments. We also analyze the trade-off between performance gains and computational overhead, and discuss which fusion configurations provide the best balance between stability and efficiency.

## 1 Introduction

With recent advancements in large language models (LLMs), the representational capacity of decoder-based architectures (Brown et al., 2020; Touvron et al., 2023a,b) has drawn increasing attention for downstream NLP tasks (Zhang et al., 2022; Sun et al., 2023). Despite their strong zero- and few-shot performance, these models are primarily trained for next-token prediction, leading to layer-wise differences in how semantic and contextual information is encoded. While final-layer embeddings are commonly used (Brown et al., 2020), prior studies in encoder-based models (Devlin et al., 2019) suggest that intermediate layers may yield richer representations for classification (Zhang et al., 2024).

The availability of generative LLMs and specialized embedding models (Lee et al., 2025; Wang et al., 2024) makes embedding fusion a viable strategy for leveraging complementary knowledge. However, practical comparisons of layer selection, pooling strategy, and multi-model fusion under a unified setting are still limited. In particular, it is still unclear how layer choice, pooling strategy, and multi-model fusion interact across generative and embedding models in text classification. To address these gaps, our contributions are as follows:

**Layer-Aware Representation Analysis** We examine how representations vary across layers and how pooling strategies affect classification performance across both embedding models and generative LLMs.

**Embedding Fusion** We show that single-layer representations do not always provide stable performance, while multi-model fusion yields more robust results.

**Adaptability in Low-Resource Settings** We show that embedding fusion is particularly effective when training data is limited, highlighting its utility in low-resource settings.

**Cost-Performance Trade-off Analysis** We analyze how the benefit of multi-model fusion changes as the number of fused models increases, and discuss which setting provides the best balance between stability and embedding extraction cost.

## 2 Related Work

### 2.1 Text Classification and Pretrained Language Models

Text classification has evolved from traditional statistical methods to deep learning-based paradigms.

*Corresponding author



*Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 38568–38577

Early research primarily utilized representations like TF-IDF and $n$-grams with models such as SVM (Pang et al., 2002) and logistic regression (Zhang et al., 2003). The paradigm shifted significantly with pre-trained language models. Prominent models like BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) capture rich contextual information through bidirectional architectures and large-scale pre-training. Fine-tuning these models on downstream tasks consistently yields superior performance compared to traditional approaches (Youngmin et al., 2024).

### 2.2 Extended Applications of LLM

The evolution of Large Language Models (LLMs) based on decoder architectures has enabled zero-shot (Kojima et al., 2023) and few-shot (Zhang et al., 2022) learning for downstream tasks such as text classification, spurring research on prompt-based approaches such as chain-of-thought (CoT) (Wei et al., 2023) and CARP (Sun et al., 2023). Numerous studies have reported on the performance of LLMs in classification tasks, including models such as GPT-3 (Brown et al., 2020) and the LLaMA series (Touvron et al., 2023a,b), as well as empirical studies analyzing their behavior (Sarkar et al., 2023; Gretz et al., 2023). However, these methods exhibit considerable performance variability depending on prompt design (Cao et al., 2024; He et al., 2024).

Meanwhile, there is growing interest in using LLMs not only as generative models but also as providers of high-quality embeddings (Tao et al., 2025). Recent research suggests that relatively lightweight LLMs (e.g. up to 7B parameters) can produce strong embedding quality with efficient computational resources (Wang et al., 2024; Lee et al., 2025). Furthermore, several studies have emphasized that different embedding layers within a model can produce optimal representations for tasks, with a particular focus on the importance of layer selection (Zhang et al., 2024). These findings highlight the role of intermediate representations in understanding the encoding behavior of transformer models in various applications (Skean et al., 2024).

### 2.3 Embedding Fusion

As diverse pretrained models, including large language models (LLMs), continue to emerge, there has been increasing interest in combining embeddings extracted from multiple models (Shinnou et al., 2018; Blandfort et al., 2019). Previous studies have reported performance improvements on tasks such as text classification and sentiment analysis through embedding fusion.

For example, LLMEmbed (Liu et al., 2024) demonstrates that combining embeddings from LLaMA2 (Touvron et al., 2023b) with those from BERT and RoBERTa can effectively leverage the distinctive representational characteristics of each model. Moreover, a variety of fusion strategies have been proposed not only in NLP, but also in other domains for instance, QUARC (Kumar et al., 2020) applies quaternion-based operations.

However, significant performance differences arise depending on the fusion method employed, and not all combinations lead to consistent improvements (Ko et al., 2024). Furthermore, since each embedding layer possesses a different representational capacity, understanding the layer-wise characteristics is critical for effective fusion (Kaushik et al., 2024).

## 3 Methodology

In this study, we investigate embedding-based text classification from three complementary perspectives. First, we revisit whether previously reported layer-wise differences in the representations of large language models also be observed in text classification tasks and further examine whether similar patterns can be observed in embedding models.

Second, we assess the stability of performance across layers and show that this limitation can be mitigated through multi-model embedding fusion.

Finally, we investigate the effectiveness of embedding fusion in low-resource settings, analyzing whether the combined representations provide a significant performance boost when labeled data is limited. We aim to determine if fusion strategies can offer a practical, data-efficient solution for robust text classification. Our experiments employ both pretrained generative LLMs (up to 14B parameters) and specialized embedding models.

### 3.1 Layer and Pooling Strategies

**Layer-wise Representation** Previous research has shown that different layers of decoder-based LLMs capture different aspects of semantic and contextual information. In this section, we verify that these results apply equally to LLMs with different model architectures, and also to embedding models explicitly optimized for representation

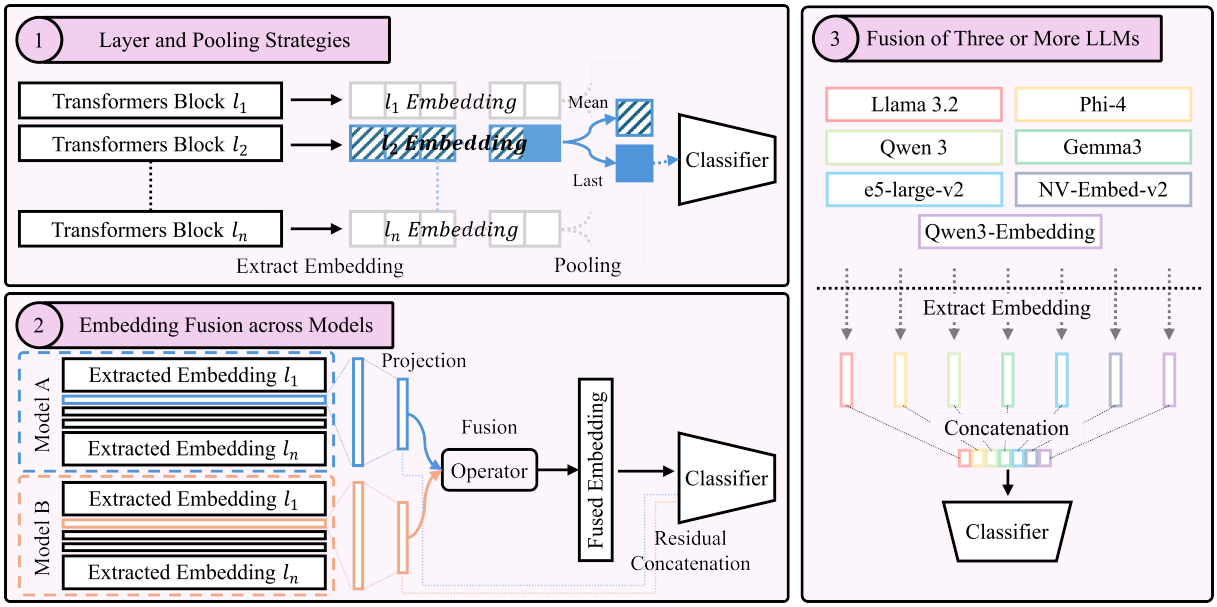


Figure 1: Overview of the experimental Desing for evaluating embedding fusion. Here, $l_n$ denotes the Transformer block layer, where $n$ represents the final layer of each model. (1) For each model, embeddings extracted from every layer are aggregated using both mean and last pooling strategies and evaluated through a classifier to measure layer/pooling sensitivity. (2) Embeddings from different models are projected onto a shared dimensional space and fused through a predefined operator (e.g., concatenation, summation, or convolution) to produce unified representations. (3) All selected embeddings are concatenated, and the fused representations are classified to obtain the final performance for multi-model fusion analysis.

learning. To assess the layer-by-layer representational power for each text data set, we systematically evaluate embeddings extracted from all layers. This quantitative analysis, conducted on both generative LLMs and dedicated embedding models, validates the findings of previous studies (Skean et al., 2024).

**Pooling** Next, we examine how pooling strategies interact with layer selection. While embedding models explicitly define pooling strategies(Tang and Yang, 2024), generative models typically do not account for pooling when extracting embeddings. Thus, choosing an appropriate pooling strategy becomes essential. In particular, we compare last pooling, which directly uses the representation at the final token position, with mean pooling, which averages hidden states across the entire sequence. For each model and dataset, we systematically analyze whether the impact of pooling remains consistent across datasets within the same model and whether performance differences emerge.

### 3.2 Embedding Fusion across Models

**Dimension Projection** Embeddings extracted from multiple models may have different dimensions, so we apply a linear projection to unify them before fusion. Specifically, to project an embedding $E \in \mathbb{R}^{d_1}$ to a target dimension $d_2$, we use learnable parameters: a weight matrix $W \in \mathbb{R}^{d_2 \times d_1}$ and a bias vector $b \in \mathbb{R}^{d_2}$. These projected embeddings are then integrated using various fusion techniques.

The choice of $d_2$ involves a trade-off: larger dimensions preserve more information but increase computational cost, while smaller ones reduce overhead at the risk of information loss. In this study, we project onto the smaller dimension to improve efficiency.

**Fusion Methods** A key focus of this study is to enhance the representational capabilities of various models by combining embeddings extracted from different LLMs. While earlier sections analyze performance across different layers, here we fix the embeddings to those from the last layer in order to ensure consistency across models and to isolate the effect of different fusion strategies. Formally, we define the fusion process as a mapping

$$F : \{E_1, E_2, \ldots, E_n\} \rightarrow E', \quad (1)$$

where $E_i \in \mathbb{R}^d$ are embeddings extracted from different models and $E' \in \mathbb{R}^{d'}$ is the fused embedding. The specific instantiations of $F$ explored in this study are as follows:

- **Concatenation:** Directly concatenating the embedding vectors $E$ along the matrix dimension to

form a single embedding:

$$E' = [E_1 \parallel E_2 \parallel \ldots \parallel E_n] \tag{2}$$

- **Sum:** After aligning the dimensions of the embeddings, we form a new embedding by element-wise addition:

$$E' = \sum_{i=1}^{n} f(E_i) \tag{3}$$

- **Multiplication:** After aligning dimensions, embeddings are reshaped into 2D arrays of size $\sqrt{d} \times \sqrt{d}$ (where $d$ must be a perfect square), and then combined via matrix multiplication:

$$\begin{aligned} E_1 \in \mathbb{R}^d &\rightarrow \textit{reshape} \rightarrow E_1' \in \mathbb{R}^{\sqrt{d}\times\sqrt{d}} \\ E_2 \in \mathbb{R}^d &\rightarrow \textit{reshape} \rightarrow E_2' \in \mathbb{R}^{\sqrt{d}\times\sqrt{d}} \\ E' &= E_1' \cdot E_2' \\ E'' &= \textit{flatten}(E') \in \mathbb{R}^d \end{aligned} \tag{4}$$

- **Hadamard (Element-wise Product):** Embeddings are combined by multiplying corresponding elements at the same position across models.

- **Quaternion Fusion** (Kumar et al., 2020): Embeddings are treated as quaternion-valued vectors and fused using quaternion operations, preserving multidimensional inter-model relationships.

- **Residually Enhanced Fusion** (Gardias et al., 2020): The newly generated embedding is combined with the original one via residual connections to incorporate additional information while preserving the original expressiveness.

**Layer-Aware Fusion** Beyond the choice of fusion techniques, we further investigate whether selecting embeddings from dataset-specific optimal layers can enhance fusion performance. For each dataset, we first identify the most effective layers using the methodology described in Section 3.1. These layers are then used in fusion experiments involving single or multiple models. We compare the classification performance of these optimal-layer-based fusions against baselines that use default or last-layer embeddings. This analysis evaluates whether tailoring layer selection to each dataset improves accuracy and stability without requiring fine-tuning of the underlying models.

Table 1: Summary of experimental setups. The top section lists dataset statistics and the bottom section details the generation and embedding models used, including dimension (Dim), layers, and parameter size (Params).

| **Section 1: Dataset Statistics** | | | |
|---|---|---|---|
| **Dataset** | **Classes** | **Train** | **Test** |
| SST-2 | 2 | 67,349 | 872 |
| MR | 2 | 40,000 | 10,000 |
| R8 | 8 | 5,485 | 2,189 |
| R52 | 52 | 6,532 | 2,568 |
| Rotten Tomatoes | 2 | 8,530 | 1,066 |
| LegalBench | 3 | 200 | 50 |
| **Section 2: Model Specifications** | | | |
| **Model Type** | **Dim** | **Layers** | **Params** |
| *Generation* | | | |
| `llama3.2` | 4096 | 28 | 3B |
| `qwen3` | 4096 | 36 | 8B |
| `gemma3` | 2304 | 48 | 12B |
| `phi4` | 3584 | 40 | 14.7B |
| *Embedding* | | | |
| `e5-large-v2` | 1024 | 24 | 0.335B |
| `nv-embed-v2` | 4096 | 32 | 7.1B |
| `qwen3-embedding` | 4096 | 24 | 8B |

### 3.3 Fusion of Three or More LLMs

While the previous sections focused on combining embeddings from two models, this section extends the analysis to three or more models. In addition to performance and stability, we also examine the added cost of larger fusion sets, primarily in terms of embedding extraction time. This allows us to assess whether the added benefit of using more models is cost-effective.

The motivation for combining multiple models is to leverage their complementary strengths. In particular, we investigate whether combining three or more models improves performance and reduces its variance, thereby leading to more stable results. The fusion method used in this experiment is primarily concatenation, and all possible combinations are tested. To reduce external sources of variation, embeddings are fixed to the last layer, and mean pooling is applied consistently across all models.

## 4 Experiment

We evaluated our approach using six benchmark datasets: SST-2 (Socher et al., 2013), MR (Maas et al., 2011), and Rotten Tomatoes (Pang and Lee, 2005) for sentiment classification, R8 and R52 for topic classification, and a subset of LegalBench (Guha et al., 2023). Following the official guidelines, LegalBench was split into training and

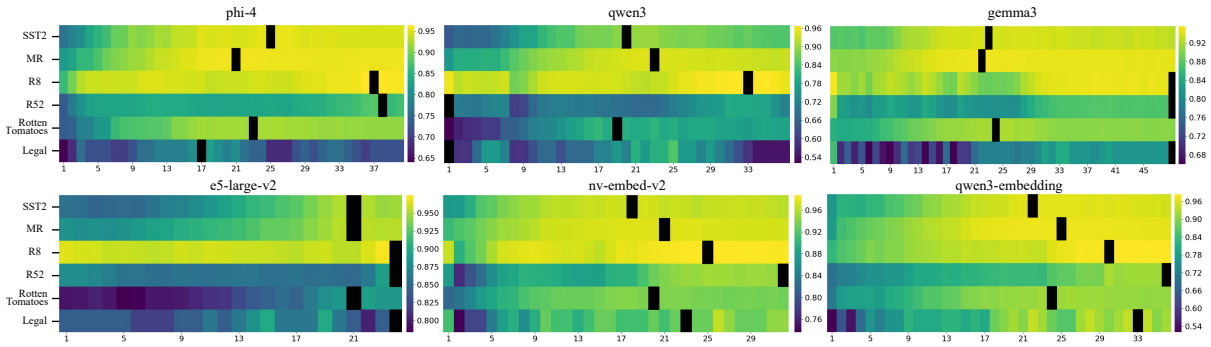


Figure 2: Comparison of Single-Layer Embedding Classification Performance in LLMs

| | Model | Layer | SST2 | | MR | | R8 | | R52 | | Rotten Tomatoes | | LegalBench | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Acc | $F_1$ | Acc | $F_1$ | Acc | $F_1$ | Acc | $F_1$ | Acc | $F_1$ | Acc | $F_1$ |
| Generation | llama3.2 | $last_{28}$ | 0.911 | 0.911 | 0.957 | 0.957 | 0.975 | 0.925 | 0.902 | 0.366 | 0.895 | 0.895 | 0.836 | 0.575 |
| | | $best_l$ | $0.950_{14}$ | $0.950_{14}$ | $0.965_{14}$ | $0.965_{14}$ | $0.979_{24}$ | $0.941_{24}$ | $0.924_{26}$ | $0.485_{26}$ | $0.910_{13}$ | $0.910_{13}$ | $0.920_{0}$ | $0.833_{15}$ |
| | phi4 | $last_{40}$ | 0.944 | 0.944 | 0.960 | 0.960 | 0.978 | 0.938 | 0.908 | 0.406 | 0.909 | 0.909 | 0.860 | 0.592 |
| | | $best_l$ | $0.954_{25}$ | $0.954_{25}$ | $0.966_{21}$ | $0.966_{21}$ | $0.980_{39}$ | $0.943_{13}$ | $0.933_{30}$ | $0.583_{30}$ | $0.923_{21}$ | $0.923_{21}$ | $0.960_{19}$ | $0.912_{19}$ |
| | qwen3 | $last_{36}$ | 0.943 | 0.943 | 0.958 | 0.958 | 0.967 | 0.867 | 0.858 | 0.196 | 0.911 | 0.911 | 0.780 | 0.535 |
| | | $best_l$ | $0.951_{24}$ | $0.951_{24}$ | $0.965_{22}$ | $0.965_{22}$ | $0.975_{32}$ | $0.916_{30}$ | $0.896_{0}$ | $0.389_{0}$ | $0.919_{25}$ | $0.919_{25}$ | $0.920_{0}$ | $0.634_{24}$ |
| | gemma3 | $last_{48}$ | 0.953 | 0.953 | 0.959 | 0.959 | 0.976 | 0.927 | 0.920 | 0.480 | 0.923 | 0.923 | 0.888 | 0.764 |
| | | $best_l$ | $0.964_{23}$ | $0.964_{23}$ | $0.967_{21}$ | $0.967_{21}$ | $0.978_{48}$ | $0.931_{48}$ | $0.922_{48}$ | $0.492_{48}$ | $0.926_{26}$ | $0.926_{26}$ | $0.900_{48}$ | $0.778_{48}$ |
| Embedding | e5-large-v2 | $last_{24}$ | 0.945 | 0.945 | 0.953 | 0.953 | 0.973 | 0.922 | 0.895 | 0.357 | 0.891 | 0.891 | 0.916 | 0.629 |
| | | $best_l$ | $0.953_{21}$ | $0.953_{21}$ | $0.956_{21}$ | $0.956_{21}$ | $0.974_{24}$ | $0.931_{24}$ | $0.896_{24}$ | $0.365_{24}$ | $0.894_{21}$ | $0.894_{21}$ | $0.920_{24}$ | $0.632_{24}$ |
| | nv-embed-v2 | $last_{31}$ | 0.955 | 0.955 | 0.967 | 0.967 | 0.983 | 0.954 | 0.951 | 0.667 | 0.922 | 0.922 | 0.940 | 0.896 |
| | | $best_l$ | $0.969_{17}$ | $0.969_{17}$ | $0.972_{20}$ | $0.972_{20}$ | $0.984_{25}$ | $0.958_{22}$ | $0.953_{31}$ | $0.681_{31}$ | $0.935_{19}$ | $0.925_{19}$ | $0.960_{22}$ | $0.912_{22}$ |
| | qwen3-embedding | $last_{36}$ | 0.949 | 0.949 | 0.964 | 0.964 | 0.982 | 0.946 | 0.950 | 0.664 | 0.906 | 0.906 | 0.920 | 0.884 |
| | | $best_l$ | $0.964_{22}$ | $0.964_{22}$ | $0.972_{25}$ | $0.972_{25}$ | $0.983_{30}$ | $0.952_{31}$ | $0.952_{36}$ | $0.699_{36}$ | $0.922_{24}$ | $0.922_{24}$ | $0.960_{33}$ | $0.911_{33}$ |

Table 2: Classification performance using embeddings extracted from specific layers. Each dataset reports both Accuracy and F1-score. Results for the last layer are averaged over five runs, while results for the best layer correspond to the best performance observed across the five runs.

test sets at an 8:2 ratio, while the original partitions were used for the other datasets.Detailed statistics of the datasets and the models employed are summarized in Table 1. The experiments involved a diverse range of generation and embedding models, including Llama3.2 (Grattafiori et al., 2024), Phi-4 (Abdin et al., 2024), Qwen3 (Yang et al., 2025), and Gemma3 (Team et al., 2025) as generative models, alongside e5-large-v2 (Wang et al., 2024), NV-Embed-v2 (Lee et al., 2025), and Qwen3-Embedding (Zhang et al., 2025). Specific configurations for each model, such as embedding dimensions ($Dim$), layer counts, and parameter scales ($Param$), are also detailed in the table.

Experiments were conducted on a system equipped with two NVIDIA RTX 4090 GPUs (24 GB memory each). A multilayer perceptron (MLP)-based classifier was applied for text classification with fused embedding vectors. Training was performed with batch size = 1024, learning rate=1e-4, optimizer=Adam, and 120 epochs. Performance was evaluated using accuracy and F1-score.

### 4.1 Results by Layer Strategies

**Layer-wise Representation** Previous research has reported that embeddings extracted from intermediate layers often exhibit stronger representational power than those from the final layer. Building on the observation that generative language models become increasingly specialized for text generation in their final layers, we conducted a detailed quantitative evaluation to examine whether this pattern consistently holds across different layers and models. As shown in Figure 2, in some cases, generative and embedding models attained maximum performance in intermediate layers rather than in the final layer. However, this trend cannot be generalized. For example, in the case of Gemma3, despite being a generative model, the final layer performed best on three datasets. Interestingly, for Qwen3, the 0th layer achieved the highest accuracy on two datasets. Moreover, even in embedding models explicitly optimized for representation learning, the final layer did not always guarantee superior performance. Table 2 presents

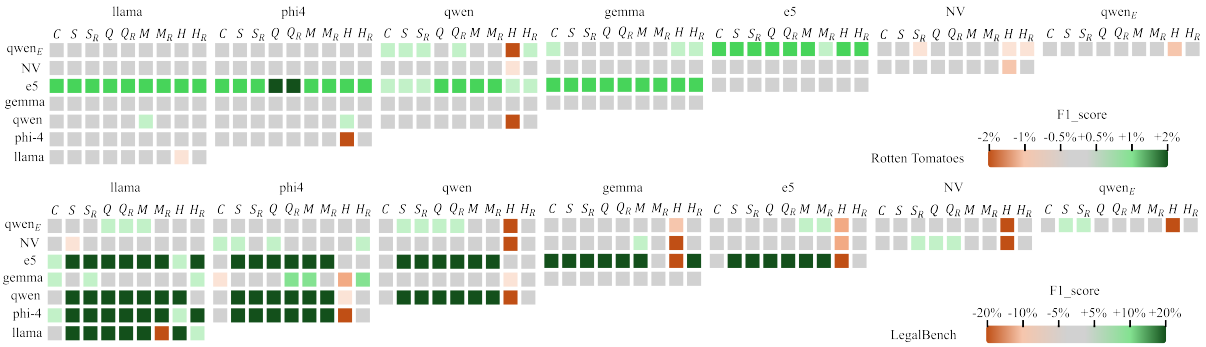


Figure 3: F1-score variations across layer combinations when combining two models on the Rotten Tomatoes and LegalBench datasets. The model abbreviations are as follows: llama(LLaMA-3.2), phi4(Phi-4), qwen(Qwen3), gemma(Gemma-12B-it), e5(e5-large-v2), NV(NV-Embed-v2), and qwen$_E$(Qwen3-Embedding). Fusion methods are denoted as C (Concatenation), S(Sum), S$_R$(Sum Residual), Q (Quaternion), Q$_R$(Quaternion Residual), M(Multiplication), M$_R$(Multiplication Residual), H(Hadamard), and H$_R$(Hadamard Residual).

| | SST2 | | MR | | R8 | | R52 | | Rotten Tomatoes | | Legal Bench | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| llama 3.2 | 0.942 | 0.942 | 0.952 | 0.952 | 0.962 | 0.881 | 0.873 | 0.291 | 0.895 | 0.895 | 0.812 | 0.554 |
| | 0.911 | 0.911 | 0.957 | 0.957 | 0.975 | 0.925 | 0.902 | 0.366 | 0.877 | 0.877 | 0.836 | 0.575 |
| phi-4 | 0.944 | 0.944 | 0.955 | 0.955 | 0.960 | 0.883 | 0.860 | 0.247 | 0.909 | 0.909 | 0.720 | 0.489 |
| | 0.930 | 0.930 | 0.960 | 0.960 | 0.978 | 0.938 | 0.908 | 0.406 | 0.885 | 0.885 | 0.860 | 0.592 |
| qwen3 | 0.943 | 0.943 | 0.954 | 0.954 | 0.938 | 0.703 | 0.820 | 0.158 | 0.911 | 0.911 | 0.708 | 0.484 |
| | 0.917 | 0.917 | 0.958 | 0.958 | 0.967 | 0.867 | 0.858 | 0.196 | 0.885 | 0.885 | 0.780 | 0.535 |
| gemma3 | 0.953 | 0.953 | 0.958 | 0.958 | 0.958 | 0.874 | 0.871 | 0.305 | 0.923 | 0.923 | 0.780 | 0.547 |
| | 0.928 | 0.928 | 0.959 | 0.959 | 0.976 | 0.927 | 0.920 | 0.480 | 0.892 | 0.892 | 0.880 | 0.764 |
| e5-large -v2 | 0.937 | 0.937 | 0.949 | 0.949 | 0.971 | 0.936 | 0.889 | 0.336 | 0.891 | 0.891 | 0.808 | 0.555 |
| | 0.946 | 0.946 | 0.953 | 0.953 | 0.973 | 0.922 | 0.895 | 0.357 | 0.882 | 0.882 | 0.916 | 0.629 |
| nv-embed -v2 | 0.949 | 0.949 | 0.962 | 0.962 | 0.977 | 0.939 | 0.931 | 0.546 | 0.929 | 0.929 | 0.880 | 0.604 |
| | 0.955 | 0.955 | 0.967 | 0.967 | 0.984 | 0.954 | 0.952 | 0.679 | 0.923 | 0.923 | 0.940 | 0.896 |
| qwen3 embedding | 0.949 | 0.949 | 0.964 | 0.964 | 0.982 | 0.946 | 0.950 | 0.664 | 0.906 | 0.906 | 0.920 | 0.882 |
| | 0.928 | 0.928 | 0.961 | 0.961 | 0.977 | 0.932 | 0.929 | 0.524 | 0.888 | 0.888 | 0.920 | 0.884 |

Figure 4: Performance differences across pooling strategies for each model and dataset.

the numerical results, comparing accuracy and F1-scores between the final and the best-performing layers. Notably, selecting the optimal layer alone improved accuracy by up to 4% and F1-score by as much as 20%.

**Pooling** Experimental results for pooling strategies across models and datasets are presented in Figure 4. Each block represents: $\frac{\text{last}/acc \quad \text{last}/F_1}{\text{mean}/acc \mid \text{mean}/F_1}$ Here, **last** refers to pooling from the final **token position** along the sequence dimension, not to the last layer of the model. During the experiments, tokenizer padding positions were fixed according to each model's baseline configuration, and only the pooling strategy was varied. To ensure a fair comparison, all experiments used the final-layer representations. Green blocks indicate the pooling method that achieved higher performance, while purple blocks represent cases where the performance gap between pooling strategies exceeded 6%. Orange blocks highlight unusual cases in which the choice of pooling method led to different trends between F1-score and accuracy.

On average, more than a 1% difference in performance was observed between pooling strategies, and in the LegalBench dataset, the gap exceeded 20%. The most notable observation arises from the distributional differences in generative models. Despite having distinct architectures, the four generative models did not exhibit model-specific consistency. Instead, the effect of pooling tended to generalize across datasets rather than across models. In contrast, embedding models showed consistent behavior across model types.

## 4.2 Fusion Strategy Experiments

This section evaluates various embedding fusion strategies using the models introduced earlier. The performance gain or loss of each fused embedding was assessed by comparing it with the higher F1-score obtained from the two individual models using their single-layer embeddings.

Figure 3 presents the results of model combinations on two datasets, Rotten Tomatoes, where accuracy has not yet saturated, and LegalBench, which suffers from limited data resources and shows instability in both accuracy and F1-score. As expected, simple concatenation between identical models did not lead to any improvement in either dataset. In the Rotten Tomatoes dataset, embedding fusion with the e5 model achieved more than a 0.5% performance gain in all cases except for combinations involving NV embeddings. Other

Table 3: Performance comparison between single-model baselines and two-model fusion methods across six text classification datasets. $l$ indicates the optimal layer index for each dataset.

| Dataset | Models | Layer | Fusion | ACC | $F_1$ | $\Delta(\%)$ |
|---|---|---|---|---|---|---|
| SST2 | NV-Embed-v2 | – | – | 0.955 | 0.955 | |
| | NV-Embed-v2, e5-large-v2 | 17, 21 | Hadamard(R) | 0.9748 | 0.9748 | +1.98% |
| MR | Qwen3-Embedding | – | – | 0.964 | 0.964 | |
| | NV-Embed-v2, e5-large-v2 | 20, 21 | Hadamard | 0.9738 | 0.9738 | +0.98% |
| R8 | NV-Embed-v2 | – | – | 0.983 | 0.954 | |
| | NV-Embed-v2, Qwen3-Embedding | 25, 30 | concatenation | 0.9863 | 0.9607 | +0.33%, +0.67% |
| R52 | NV-Embed-v2 | – | – | 0.951 | 0.667 | |
| | NV-Embed-v2, Qwen3-Embedding | 31, 36 | Hadamard(R) | 0.9599 | 0.7702 | +0.82%, +10.32% |
| Rotten Tomatoes | NV-Embed-v2 | – | – | 0.922 | 0.922 | |
| | NV-Embed-v2, gemma3-12b-it | 19, 26 | Sum(R) | 0.9371 | 0.9371 | +1.51% |
| LegalBench | Qwen3-Embedding | – | – | 0.940 | 0.896 | |
| | NV-Embed-v2, e5-large-v2 | 22, 24 | Multiplication(R) | 0.9800 | 0.9270 | +4.0%, +3.1% |

Table 4: Classification accuracy and F1-score with multi-model fusion (≥3 models). $N(C)$ denotes the number of possible combinations of $n$ models selected from the seven models, without repetition or order. All models use embeddings extracted from the last layer with mean pooling.

| Models | SST2 | | MR | | R8 | | R52 | | Rotten Tomatoes | | LegalBench | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc | $F_1$ | Acc | $F_1$ | Acc | $F_1$ | Acc | $F_1$ | Acc | $F_1$ | Acc | $F_1$ |
| Single | 0.911-0.955 | 0.911-0.955 | 0.953-0.967 | 0.953-0.967 | 0.967-0.983 | 0.916-0.954 | 0.858-0.951 | 0.196-0.667 | 0.891-0.922 | 0.891-0.922 | 0.780-0.940 | 0.535-0.896 |
| 3(35) | 0.930-0.961 | 0.930-0.961 | 0.961-0.971 | 0.961-0.971 | 0.978-0.984 | 0.932-0.956 | 0.921-0.956 | 0.461-0.721 | 0.891-0.925 | 0.891-0.924 | 0.860-0.980 | 0.592-0.927 |
| 4(35) | 0.934-0.963 | 0.934-0.963 | 0.961-0.971 | 0.961-0.971 | 0.978-0.985 | 0.932-0.958 | 0.934-0.957 | 0.545-0.720 | 0.894-0.925 | 0.894-0.925 | 0.880-0.98 | 0.620-0.927 |
| 5(21) | 0.941-0.962 | 0.941-0.962 | 0.962-0.971 | 0.962-0.971 | 0.980-0.984 | 0.942-0.955 | 0.939-0.956 | 0.589-0.722 | 0.899-0.925 | 0.899-0.925 | 0.920-0.980 | 0.884-0.927 |
| 6(7) | 0.956-0.963 | 0.956-0.963 | 0.965-0.971 | 0.965-0.971 | 0.982-0.984 | 0.945-0.953 | 0.946-0.956 | 0.639-0.724 | 0.906-0.925 | 0.906-0.925 | 0.940-0.980 | 0.898-0.927 |
| 7(1) | 0.958 | 0.958 | 0.971 | 0.971 | 0.982-0.983 | 0.949-0.951 | 0.956 | 0.697 | 0.924-0.925 | 0.924-0.925 | 0.940-0.980 | 0.898-0.927 |

combinations also showed slight improvements, but in most cases, the performance increase was not substantial.

These findings suggest that, in embedding fusion, the improvement in classification performance arises not merely from the inherent strength of a single model but from the integration of models with distinct representational spaces, which produces richer and more stable feature representations. Meanwhile, in the LegalBench dataset, despite its unstable distribution, most fusion cases exhibited over 20% improvement in F1-score, highlighting that the fusion effect becomes more pronounced under limited-data conditions.

### 4.3 Optimal-Layer Fusion Analysis

According to the previous experiments, we confirmed that selecting an appropriate embedding layer alone can improve classification performance, and that combining embeddings from different models can further enhance it. However, since the optimal embedding layer varies depending on dataset characteristics and model architecture, it is practically infeasible to universally determine and fuse the optimal layers in real-world applications. Therefore, in this section, we assume that the optimal layer of each model is known in advance and analyze the theoretical upper bound of performance improvement that embedding fusion can achieve under such ideal conditions.

Table 3 presents the results of combining the two embeddings that achieved the highest performance for each dataset, compared with the best-performing single embedding. Although the observed improvements are marginal, these results suggest that layer-aware embedding fusion can potentially overcome the performance plateau inherent to single-embedding models.

### 4.4 Fusion of Three or More Models: Stability and Cost Trade-off

We extend our analysis to the fusion of three or more models to examine how performance changes as the number of integrated representations increases. For each configuration, we report the minimum and maximum accuracy and F1 scores, and further analyze the trade-off between performance and cost.

Table 4 summarizes the results for all multi-model fusion settings, where the number in parentheses indicates the total number of possible model

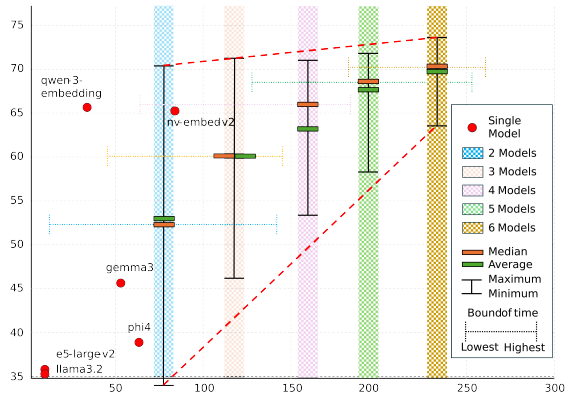


Figure 5: Trade-off Analysis between Inference Latency and Performance Stability

combinations. The "Single" row represents the range of performance observed from individual models. As the number of fused models increases, both accuracy and F1 score generally improve across datasets, suggesting that integrating heterogeneous embeddings leads to more stable performance.

Here, stability is reflected mainly in the increase of the minimum performance rather than in the maximum performance. This effect is most pronounced in the R52 and LegalBench datasets, where the minimum F1 score increases substantially from 0.196 to 0.697 and from 0.535 to 0.898, respectively. Similar patterns are observed across the remaining datasets, showing that the lower bound of performance shifts upward with larger fusion sets.

Figure 5 further illustrates the trade-off between embedding extraction time and F1 score on the R52 dataset. Among the tested settings, the three-model fusion provides the most favorable balance between performance gain and cost. It achieves the largest improvements in both minimum and maximum performance. By contrast, the four-model setting further stabilizes performance, but its gain in maximum performance is relatively small, suggesting diminishing returns beyond three models.

## 5 Conclusion

Previous studies have shown that language models have different layer-wise representations, but it is still unclear whether these differences consistently help text classification across both generative and embedding models. In addition, it is not clear whether there is a single pooling strategy that works best in all cases.

Our analysis shows that some layers contain more discriminative features, but this does not always lead to better classification results. In some cases, earlier layers perform best. We also find that, in generative models, the effect of pooling depends more on the dataset than on the model itself.

Based on these findings, we show that multi-model embedding fusion makes performance more stable by reducing sensitivity to layer and pooling choices. This effect is especially pronounced in low-resource settings. We also find that larger fusion sets can improve performance, but they also increase embedding extraction cost. In our setting, three-model fusion gives the best balance between performance, stability, and cost. Overall, our results show the practical utility of layer-aware multi-model fusion for text classification.

## 6 Limitation

While this study provides a systematic analysis of layer selection, pooling strategies, and embedding fusion, several limitations should be acknowledged. First, our investigation was limited to text classification tasks. Future work could expand this scope to other downstream objectives such as retrieval, clustering, or text generation, offering a more comprehensive view of how embedding fusion behaves across different task types. Second, the proposed investigation investigation focused on static, task-agnostic fusion rather than incorporating adaptive or learnable weighting mechanisms that could dynamically optimize contributions from multiple models. Finally, although our findings demonstrate the effectiveness of embedding fusion in low-resource settings, a more extensive evaluation on domain-shifted and noisy data is needed to further validate its robustness and generalization.

## 7 Future Work

Future research may explore adaptive fusion strategies that dynamically modulate the contribution of each model or layer based on task characteristics and data properties. Another promising direction is to extend layer-aware embedding fusion to multimodal contexts by integrating textual, visual, and structured representations, thereby enhancing representational diversity and transferability. In addition, investigating how the efficiency of embedding fusion scales with model size and architecture diversity could offer valuable insights into the theoretical and practical limits of representation integration.

## 8 Acknowledgements

We thank the anonymous reviewers and the area chair for their constructive comments. This research was supported by the Korea Institute of Science and Technology Information (KISTI) and the IITP (Institute of Information & Communications Technology Planning & Evaluation)-ICAN (ICT Challenge and Advanced Network of HRD) grant funded by the Korea government (Ministry of Science and ICT) (IITP-2026-RS-2022-00156394).

## References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, and 8 others. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

Philipp Blandfort, Tushar Karayil, Federico Raue, Jörn Hees, and Andreas Dengel. 2019. Fusion strategies for learning user embeddings with neural networks. *Preprint*, arXiv:1901.02322.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Bowen Cao, Deng Cai, Zhisong Zhang, Yuexian Zou, and Wai Lam. 2024. On the worst prompt performance of large language models. *Preprint*, arXiv:2406.10248.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Przemek Gardias, Eric Arthur, and Huaming Sun. 2020. Enhanced residual networks for context-based image outpainting. *Preprint*, arXiv:2005.06723.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Shai Gretz, Alon Halfon, Ilya Shnayderman, Orith Toledo-Ronen, Artem Spector, Lena Dankin, Yannis Katsis, Ofir Arviv, Yoav Katz, Noam Slonim, and Liat Ein-Dor. 2023. Zero-shot topical text classification with LLMs - an experimental study. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9647–9676, Singapore. Association for Computational Linguistics.

Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, Aditya Narayana, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel N. Rockmore, Diego Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory M. Dickinson, Haggai Porat, Jason Hegland, and 21 others. 2023. Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models. *Preprint*, arXiv:2308.11462.

Jia He, Mukund Rungta, David Koleczek, Arshdeep Sekhon, Franklin X. Wang, and Sadid Hasan. 2024. Does prompt formatting have any impact on llm performance? *Preprint*, arXiv:2411.10541.

Arjun Ramesh Kaushik, Sunil Rufus R. P, and Nalini Ratha. 2024. Enhancing authorship attribution through embedding fusion: A novel approach with masked and encoder-decoder language models. *Preprint*, arXiv:2411.00411.

Young Su Ko, Jonathan Parkinson, and Wei Wang. 2024. Benchmarking text-integrated protein language model embeddings and embedding fusion on diverse downstream tasks. *bioRxiv*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *Preprint*, arXiv:2205.11916.

Deepak Kumar, Nalin Kumar, and Subhankar Mishra. 2020. Quarc: Quaternion multi-modal fusion architecture for hate speech classification. *Preprint*, arXiv:2012.08312.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2025. Nv-embed: Improved techniques for training llms as generalist embedding models. *Preprint*, arXiv:2405.17428.

Chun Liu, Hongguang Zhang, Kainan Zhao, Xinghai Ju, and Lin Yang. 2024. Llmembed: Rethinking lightweight llm's genuine function in text classification. *Preprint*, arXiv:2406.03725.

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. *Preprint*, arXiv:1907.11692.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. *Preprint*, arXiv:cs/0205070.

Souvika Sarkar, Dongji Feng, and Shubhra Kanti Karmaker Santu. 2023. Zero-shot multi-label topic inference with sentence encoders and LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16218–16233, Singapore. Association for Computational Linguistics.

Hiroyuki Shinnou, Xinyu Zhao, and Kanako Komiya. 2018. Domain adaptation using a combination of multiple embeddings for sentiment analysis. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*, Hong Kong. Association for Computational Linguistics.

Oscar Skean, Md Rifat Arefin, Yann LeCun, and Ravid Shwartz-Ziv. 2024. Does representation matter? exploring intermediate layers in large language models. *Preprint*, arXiv:2412.09563.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Xiaofei Sun, Xiaoya Li, Jiwei Li, Fei Wu, Shangwei Guo, Tianwei Zhang, and Guoyin Wang. 2023. Text classification via large language models. *Preprint*, arXiv:2305.08377.

Yixuan Tang and Yi Yang. 2024. Pooling and attention: What are effective designs for llm-based embedding models? *Preprint*, arXiv:2409.02727.

Chongyang Tao, Tao Shen, Shen Gao, Junshuo Zhang, Zhen Li, Kai Hua, Wenpeng Hu, Zhengwei Tao, and Shuai Ma. 2025. Llms are also effective embedding models: An in-depth overview. *Preprint*, arXiv:2412.12591.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, and 1 others. 2023a. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. Text embeddings by weakly-supervised contrastive pre-training. *Preprint*, arXiv:2212.03533.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Lee Youngmin, Lang S. I. D. Andrew, Cai Duoduo, and Wheat R. Stephen. 2024. The role of model architecture and scale in predicting molecular properties: Insights from fine-tuning roberta, bart, and llama. *Preprint*, arXiv:2405.00949.

Haoxing Zhang, Xiaofeng Zhang, Haibo Huang, and Lei Yu. 2022. Prompt-based meta-learning for few-shot text classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1342–1357, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jian Zhang, Rong Jin, Yiming Yang, and Alexander Hauptmann. 2003. Modified logistic regression: An approximation to svm and its applications in large-scale text categorization. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, volume 2, pages 888–895.

Yang Zhang, Yanfei Dong, and Kenji Kawaguchi. 2024. Investigating layer importance in large language models. *Preprint*, arXiv:2409.14381.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.