# Exploring Pooling Strategies and Layer Aggregation in Transformer Encoders for Text Classification

Anonymous ACL submission

## Abstract

The choice of pooling strategies and layer selection and aggregation plays a crucial role in the quality of sentence embeddings in Transformer-based models for classification. While the [CLS] token is commonly used for sentence representation, research suggests that alternative pooling methods, such as token averaging, often yield better results. In this work, we systematically study various pooling techniques, including average, sum, and max pooling, as well as novel combinations, alongside layer aggregation strategies for sentence and document embeddings in Transformer encoder-only models. Additionally, we propose to the concatenation of multiple pooling methods to represent a single document. Our experiments, conducted on multiple text classification benchmarks, demonstrate that carefully selecting pooling methods and layer combinations can improve classification accuracy by up to 9% compared to standard approaches. These findings emphasize the importance of exploring diverse strategies for sentence representation and offer valuable insights for optimizing embedding extraction in NLP tasks.

## 1 Introduction

The Transformer architecture (Vaswani et al., 2017) has revolutionized NLP through contextual word representations (Lin et al., 2022). Models like BERT (Devlin et al., 2018), GPT and its successors (Radford et al., 2018, 2019; Brown et al., 2020; Achiam et al., 2023) have driven a paradigm shift, significantly improving NLP tasks (Wolf et al., 2020). Despite their success, these models produce token-level outputs, and using special tokens (e.g., [CLS] in BERT) for sentence embeddings often yields suboptimal results. Research therefore favors aggregation techniques like pooling (Li and Li, 2024; Stankevičius and Lukoševičius, 2024).

Pooling aggregates token vectors into a single sentence embedding (Xue et al., 2024), typically using mean or sum operations to retain key semantics. Performance also depends on the selected model layers (Dubey et al., 2023; Hosseini et al., 2023). However, pooling methods struggle with token order and long documents, often degrading representations. Though solutions exist, like using intermediate layers or diverse pooling, most are applied ad hoc, without systematic evaluation. This gap is critical given the variety of encoders used today, such as RoBERTa (Liu et al., 2019), DeBERTaV3 (He et al., 2021), SBERT (Reimers and Gurevych, 2019), SimCSE (Gao et al., 2021), and HNCSE (Liu et al., 2024b).

While decoder-only models dominate recent AI trends, encoder-based architectures like BERT remain the standard for embedding generation in tasks such as classification, RAG, duplicate detection, sentiment analysis, and similarity measurement. Although some studies (Fu et al., 2024) use large autoregressive LLMs for sentence encoding, their high computational cost hinders scalability. Recent work (Patil et al., 2023; Hongliu, 2024) continues to explore ways to improve text representations in encoder-only models. We argue that advancing pooling strategies and context-aware embedding training is essential to enhance sentence and document representations.

In this context, this work proposes a comprehensive systematic study of approaches for generating representative embeddings for sentences and documents based on Transfomer encoder-only models for classification tasks, focusing on four main aspects:

- **Extraction of individual token information and pooling**: Investigating ways to combine token embeddings into a single vector representation. We evaluated the use of the [CLS] token to represent sentences as well as average, sum and max pooling of token embeddings. We also study the impact of excluding special tokens, such as [CLS] and [SEP], and

stopwords from the pooling. The stopwords used were extracted from the standard English list provided by the Natural Language Toolkit - NLTK[1]), a commonly used tool in NLP research.

- **Impact of extraction layer selection and aggregation**: Analyzing the performance variations in downstream tasks when extracting embeddings from different model layers. We also conducted experiments with various layer aggregation strategies, including sum aggregation (SUM), average aggregation (AVG), and individual layers preceding the final layer.
- **Generalization across encoders**: Evaluating whether the observed trends hold when using different encoding models.
- **Combinations of representations**: Evaluate whether combining multiple pooling strategies along with layer aggregation via concatenation can produce better results.

In addition to reviewing existing pooling methods, we propose novel strategies for pooling and aggregating encoder layers to better capture token-level information. We also introduce techniques for combining multiple pooling and layer aggregation methods into a single representation.

Using the SentEval toolkit (Conneau and Kiela, 2018), we evaluate these strategies across standard text classification benchmarks. Our approach yields substantial improvements: SBERT average accuracy increased by 4.31% (85.27% to 89.58%), and DeBERTaV3 by 8%. Compared to state-of-the-art methods, our technique achieved superior average accuracy, gaining 0.9% against the best encoder-only baseline, while simply selecting specific pooling and layer aggregation options.

The remainder of this paper is organized as follows: Section 2 discusses related work; Section 3 introduces our pooling and aggregation strategies; Section 4 presents the experiments and results; and Section 5 concludes the paper.

## 2 Related Works

Sentence embeddings, which represent the meaning of a sentence as a dense vector, have been extensively studied as a fundamental task in natural language processing (NLP). Recently, leveraging pretrained language models for sentence embedding has become a dominant approach (Cha and Lee, 2024). Various techniques have been proposed to improve both feature extraction and computational efficiency, including Transformer-based architectures, pooling strategies, and contrastive learning methods.

A crucial aspect of sentence representation learning is the pooling strategy, which determines how token embeddings are aggregated to obtain a single vector representing the entire sentence. Traditional methods include mean pooling, which averages the embeddings of all tokens, and `[CLS]` pooling, which uses the `[CLS]` token's embedding as the sentence representation.

BERT (Bidirectional Encoder Representations from Transformers) introduced bidirectional pretraining and Masked Language Modeling (MLM), enabling the capture of rich contextual representations. Sentence embeddings in BERT are typically derived from the `[CLS]` token (Devlin et al., 2018). RoBERTa improved upon BERT by removing the Next Sentence Prediction (NSP) objective and adopting a more robust training strategy, enhancing token embeddings' generalization. These embeddings can be aggregated using either `[CLS]` pooling or mean pooling (averaging token embeddings) (Liu et al., 2019).

To further improve sentence representations, several model adaptations have been proposed. DeBERTaV3 introduced Replaced Token Detection (RTD) as a pretraining objective, refining contextual representations while maintaining computational efficiency. However, it still primarily relies on `[CLS]` and mean pooling for sentence embedding generation (He et al., 2021).

Contrastive learning has emerged as a promising approach for producing more discriminative sentence representations. SimCSE introduced a simple yet effective contrastive learning framework in which different forward passes of the same sentence, with dropout applied, serve as positive pairs, while other sentences in the batch act as negative samples (Gao et al., 2021). This approach optimizes sentence representations by encouraging semantically similar sentences to have closer embeddings in vector space. Building upon this idea, HNCSE incorporated hard negative mining, where semantically similar sentences belonging to different classes are used as negative samples to refine class separation and improve sentence-level discriminability (Liu et al., 2024b).

Previous works predominantly relied on the `[CLS]` token or the average of tokens from the last layer for sentence representation. While widely

[1]https://www.nltk.org/

adopted, recent studies suggest that alternative strategies can yield substantial improvements. For instance, BERT-LC (BERT Layers Combination) proposed aggregating multiple Transformer layers to capture hierarchical semantic information, outperforming methods that rely solely on the final layer (Hosseini et al., 2023). However, it aggregated layers solely by averaging them, and our experiments show that summing layers achieve superior results. Similarly, AoE (Angle-optimized Embeddings) introduced an optimization method to reduce gradient saturation and enhance class separation by refining angular differences in the embedding space (Li and Li, 2024), evaluating sentence embeddings using `[CLS]`, average, and max pooling.

The integration of layer aggregation, pooling strategies, and contrastive learning has proven to be effective in generating high-quality sentence embeddings. In this work, we explore these techniques by combining multiple pooling methods and layer aggregation strategies to enhance the quality and expressiveness of sentence representations. The next section details our proposed approach.

## 3 Proposed Approach

Given an input sentence, the goal of a sentence embedding model is to generate a vector that captures its semantic and/or syntactic information. To obtain a sentence embedding, we first pass the sentence through a Transformer model, which outputs the tensor $H$, then apply a pooling function $p$, which can be $max$, $mean$, etc (Hosseini et al., 2023).

Our approach to studying pooling involves experimenting with various vector pooling techniques combined with different hidden layer aggregation methods in Transformer encoder-only models (specifically BERT-based models). These pooling techniques have been widely explored in the literature (Hosseini et al., 2023; Reimers and Gurevych, 2019; Liu et al., 2024b), and, in this work, we introduce novel combinations of pooling strategies and layer aggregation methods.

Our work focuses on four key aspects: identifying the most effective pooling strategy, determining the best layers to use for pooling, determining the best layer combination, and evaluating the interplay between different pooling methods and layer aggregation techniques as well as their combinations. First, we assess four BERT based encoder models in our initial experiments to establish a baseline: BERT, RoBERTa, DeBERTaV3 and SBERT (in our case, using the *allmpnet* model[2]).

Second, we evaluate simple pooling methods applied to the last layer to identify the most effective strategies, including a proposed variation that excludes special tokens and stopwords. Third, we examine the impact of selecting different Transformer layers for pooling and explore layer aggregation techniques to assess their effect on sentence embeddings. Finally, we investigate the combination of pooling techniques through concatenation (using two or three token vectors) alongside layer aggregation to determine the optimal configuration for producing high-quality sentence embeddings.

Below we describe the pooling techniques and aggregation strategies used in our study.

### 3.1 Pooling Techniques

#### 3.1.1 Classification Token

Our first approach to sentence embedding is to simply use the `[CLS]` token. Encoder-only models such as BERT and its derivatives generate this token as a common method for representing a sentence as a single vector, as demonstrated in studies like (Devlin et al., 2018), among others.

#### 3.1.2 Simple Pooling and Simple-NS Pooling

Next, we propose studying the application of basic pooling techniques from the literature to conduct initial experiments. These pooling methods are listed below:

- **AVG**: Simple average of the token embeddings of a given layer from a given input text;
- **SUM**: Simple sum of the token embeddings of a given layer from a given input text;
- **MAX**: Embedding generated by selecting the maximum value across each dimension from the token embeddings of a given layer of a given input text.

All pooling methods exclude the `[PAD]` token. Additionally, we conducted experiments excluding special tokens `[CLS]` and `[SEP]` and stopwords to assess their impact on pooling. To evaluate this effect, we created three variations by removing these tokens: **AVG-NS**, **SUM-NS**, and **MAX-NS**.

#### 3.1.3 Pooling Concatenation

We also evaluated whether combining two different pooling methods could impact classification

[2]https://huggingface.co/sentence-transformers/all-mpnet-base-v2

performance by concatenating their vectors into a single representation. Since there are six pooling methods plus the [CLS] token (treated as another pooling technique for simplicity), pairwise combinations resulted in 21 new pooling strategies. These combinations are denoted by a plus sign, such as **CLS+SUM**, which represents the concatenation of the [CLS] vector and the **SUM** pooling vector.

We also explored concatenating three pooling vectors, generating 35 additional pooling configurations. These are denoted by two plus signs, such as **CLS+AVG+AVG-NS**.

All concatenation techniques produce a single vector with increased dimensionality, potentially improving classification performance. Preliminary experiments with four-vector concatenation, with 35 additional pooling configurations, showed inferior results, so they were omitted for clarity and space.

### 3.2 Hidden Layers Selection and Aggregation

#### 3.2.1 Single Hidden Layers

We evaluated applying the pooling to the *n*th encoding layer to assess the impact of extracting embeddings from different hidden layers rather than exclusively from the last layer, as commonly done in (Devlin et al., 2018) and most prior works.

This hypothesis has been explored in studies such as (Liu et al., 2024c) and (Jin et al., 2024). In this paper, we denote this technique using the suffix **LYR-X**, where $X$ is the layer number.

#### 3.2.2 Sum of Hidden Layers

Instead of applying pooling to a single layer, we apply it to the sum of token embeddings across $X$ hidden layers from the input model. The goal is to evaluate whether different layers contribute distinct information about each token. Additionally, summing layers amplifies consistently larger values with the same sign while dampening values closer to zero or those with varying signs.

In (Devlin et al., 2018), layer summation was explored, but only for token-based tasks, not for sentence representation. We conducted experiments summing between 2 and 12 consecutive layers. In our notation, summed layers are denoted by the prefix **SUM**, followed by the layer range. For example, **SUM-7-10** represents the sum of hidden layers 7, 8, 9, and 10.

#### 3.2.3 Average of Hidden Layers

Similar to summing hidden layers, we also propose aggregating layers by averaging their hidden representations. In our experiments, averaged layers are denoted by the prefix **AVG**, followed by the layer range.

## 4 Experiments and Results

In our experiments, we analyze the impact of different pooling and layer selection/aggregation strategies on text classification tasks. The goal is to identify pooling techniques that enhance representations and improve performance without modifying the encoding network structure or requiring retraining/fine-tuning. Given the numerous variations in pooling methods, we systematically evaluate different approaches based on their nature. We begin with commonly used pooling techniques from the literature, then assess their effectiveness when extracting information from different Transformer layers, followed by the impact of layer aggregation, and finally, the effects of concatenating multiple pooling methods. In our experiments, we evaluate:

- **4 different base encoder models:** BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), DeBERTa (He et al., 2021), Sentence BERT (SBERT) (Reimers and Gurevych, 2019);
- **63 differents poolings**: 7 simple poolings (CLS, AVG, AVG-NS, SUM, SUM-NS, MAX and MAX-NS), 21 poolings generated by two-vector concatenation and 35 poolings generate by three-vector concatenations;
- **144 differents layers (selected or aggregations)**: 12 hidden layers (because we use only base models), 66 aggregates by SUM and 66 aggregates by AVG.

The total number of variations is: 4 x 63 x 144 = 36,288 variation results (9,072 per encoder model). We present a subset of the results during this analysis due to the sheer amount of variations, while making considerations regarding the full set where relevant.

### 4.1 Datasets and Evaluation Setup

For text classification (i.e, transfer tasks), we used seven benchmark datasets from the popular SentEval toolkit: MR, CR, SUBJ, MPQA, SST2, TREC and MRPC. These datasets are used primarily to assess the classification performance of text embeddings and have been used in several previous works,

such as (Liu et al., 2024b), (Gao et al., 2021), (Li and Li, 2024), (Hosseini et al., 2023), and (Fu et al., 2024). To ensure a fair comparison, we follow the setup of our baselines and use the default SentEval parameters, primarily relying on the average accuracy score of test datasets for evaluation.

We generate sentence embeddings using the proposed approach (combinations of pooling techniques and selected/aggregated layers) and employ them as input vectors for a logistic regression classifier. The classifier is configured in the SentEval Toolkit (Conneau and Kiela, 2018) with the following parameters: 10-fold validation, Adam optimizer, batch size of 1,024, one training epoch and tenacity=5. We adopt this method as it is a widely used embedding evaluation strategy, also employed by most of our baselines.

### 4.2 Basic Pooling Methods

To begin our analysis, we first identify which basic pooling methods yield the best results when applied to the last embedding layer of the Transformer encoder. We focus on this layer configuration as it is the most commonly used in recent methods from the literature, with averaging token embeddings being the most popular pooling approach (Liu et al., 2024a). The results for these pooling configurations are shown in Figure 1, which presents the average performance across all SentEval benchmarks for each pooling method across the evaluated encoders.

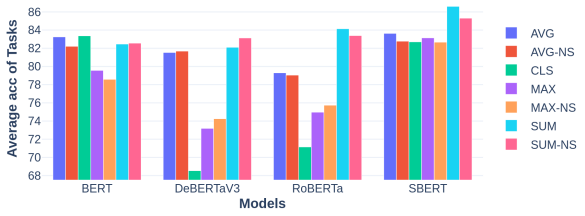



Figure 1: Average accuracy of tasks per model using basic pooling methods. The average accuracy is the mean accuracy of all eight benchmarks.

Several conclusions can be drawn from this analysis: (1) Although originally proposed for sentence-level representations, the [CLS] token consistently yielded the worst results across all encoders excluding the original BERT. This aligns with recent text embedding studies (Liu et al., 2024a), where pooling methods such as AVG generally perform better. (2) MAX pooling also performed worse than AVG and SUM, which were consistently the best choices across benchmark datasets. (3) Excluding special tokens during pooling had mixed results, improving DeBERTa slightly while having worse results for SBERT and RoBERTa. (4) SUM achieved the best results for all encoders except BERT, with SUM-NS achieving the best results for DeBERTa only. These findings were also observed in the validation set.

Based on the results from pooling embeddings in the last layer, we conclude that [CLS] and MAX-based pooling methods are not the most effective, a finding consistent with previous studies. Conversely, SUM aggregation produced surprisingly strong results despite being less commonly used in prior works compared to AVG.

### 4.3 Pooling Layer Selection Impact

Next, we evaluate the impact of pooling layer selection on classification tasks. Due to space constraints and for clarity in the figures, we focus on AVG and SUM pooling, both with and without special tokens/stopwords, as they produced the best results in layer 12 across all encoders. Although not presented, the advantage of AVG and SUM over [CLS] and MAX observed in the last layer persists in the earlier layers.

As shown in Figure 2, using the last layer (layer 12 in BERT-base-based encoders) resulted in worse performance compared to layers 7 to 10. This aligns with (Liu et al., 2024c), which suggests that the last-layer embeddings are primarily optimized for token prediction and contain weaker semantic information. Conversely, layers below 7 yielded consistently lower performance across all pooling methods. For both DeBERTa and SBERT, the best results were achieved in layers 7 through 11, with SUM pooling performing best. Regarding encoders, while SBERT kept the strongest results overall, DeBERTa had a overally superior performance when compared to RoBERTa and BERT-based encoders when considering other layers, regardless of the pooling method, with occasional ties or competitive results in some of the earlier layers, unlike when pooling from the last layer. All trends in Figure 2 were also observed in the validation set.

The best results were mostly obtained using SUM pooling from the third to forth-to-last layer (layers 9 and 10). With SBERT, SUM pooling achieved an average benchmark score of 88.02 (85.99 with SUM-NS). DeBERTa with SUM aggregation reached 87.9. For comparison, the best result using layer 12 was 86.61 (SBERT + SUM).

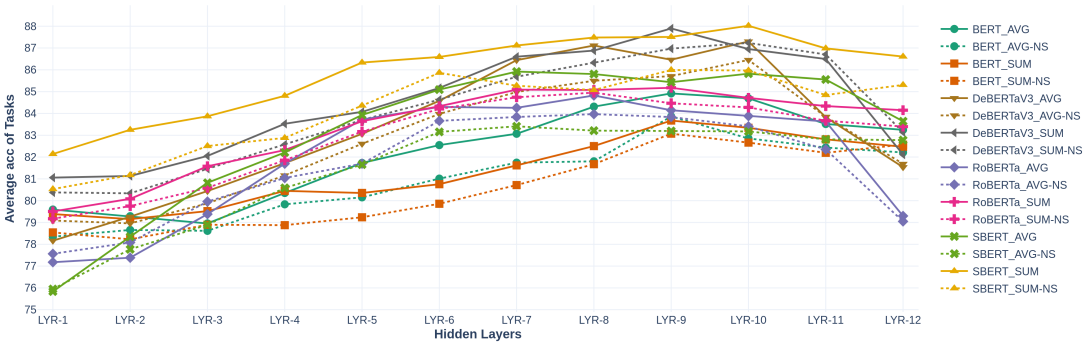



Figure 2: Average accuracy of tasks per model using the best basic pooling methods. The layers range from 1 to 12 because BERT-base models are used.

These results indicate that pooling from layers near the end, but not the last, consistently improves classification performance, differing from the standard approach in most prior studies. Given this behavior, we further explore and evaluate methods for aggregating multiple layers before pooling.

### 4.4 Aggregating Layers

Figure 3 shows results for layer aggregation prior to pooling in SBERT and DeBERTa. Since aggregation occurs at the token level, pooling is still required to obtain sentence or document embeddings. We report results for four pooling methods: AVG, AVG-NS, SUM, and SUM-NS. For each pooling method, only the best-performing aggregation size is shown, i.e., the optimal number of layers $N$ for that method, resulting in 44 lines per encoder-layer aggregation pair (11 per pooling method). Due to space constraints and the inferior performance of average aggregation, we present results for sum aggregation in SBERT (Figure 2(a)) and DeBERTa (Figure 2(b)), and average aggregation results for both models in Figure 2(c).

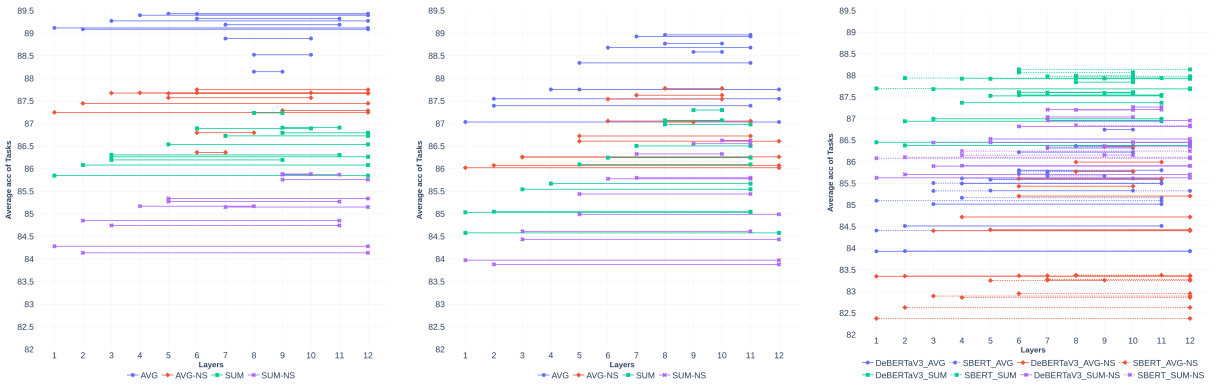
Summing layers combined with AVG pooling consistently outperformed other strategies by a large margin. On the other hand, averaging layers lead to inferior overall results, with SUM pooling having the best performance in this scenario. This advantage may stem from signal amplification: dimensions with consistently high absolute values across layers are reinforced, while inconsistent or low-variance dimensions are diminished. To our knowledge, summing layers followed by AVG pooling has not been previously explored in sentence encoding. Interestingly, using the same method for both aggregation and pooling (e.g., averaging layers with AVG pooling) reduced performance. This is notable, as prior works that applied layer aggregation (e.g., (Hosseini et al., 2023)) used average aggregation with average pooling.

For SBERT, aggregating layers 6–12 yielded the best performance, with 5–12 a close second. This aligns with the individual layer results in Figure 2, where layers 1–4 underperformed. For DeBERTa, the best results came from aggregating layers 8–11, closely followed by 7–11. The absence of layer 12, which is typically used for pooling, is notable and consistent with Figure 2, where it performed poorly for this model.

### 4.5 Two and Three vector concatenation

In addition to traditional single-token representations, we explore concatenating two or three pooled tokens to represent a document. Figure 4 presents results for these concatenation strategies. For each layer aggregation size, we report the best-performing pooling combination for DeBERTa and SBERT using two- and three-token concatenations. Four-token representations were also evaluated but consistently underperformed, and are omitted for brevity. Concatenations involving SUM and MAX pooling, as well as average layer aggregation, yielded poor results across all settings and are likewise excluded from Figure 4.

When concatenating two tokens, including the `[CLS]` token generally underperformed compared to combining AVG and AVG-NS, across both encoders. The best two-token configuration was DeBERTa-AVG+AVG-NS with sum aggregation over layers 8–10, achieving an average accuracy of 89.4. This performance was nearly tied with the best three-token setup—CLS+AVG+AVG-NS using layers 6–11 for SBERT and 8–10 for DeBERTa, both reaching 89.65. These multi-token results surpass the best single-token configuration (SBERT-AVG/SUM, layers 6–12) which achieved




(a) SBERT-sum (b) DeBERTaV3-sum (c) SBERT/DeBERTaV3-avg

Figure 3: Average accuracy of DeBERTa and SBERT across different layer aggregation strategies (sum and average of embeddings) and pooling methods (AVG, AVG-NS, SUM, SUM-NS). The x-axis indicates the layers being aggregated, and the y-axis shows the corresponding average accuracy. For each combination of aggregation size and pooling method, only the highest-performing result is shown.

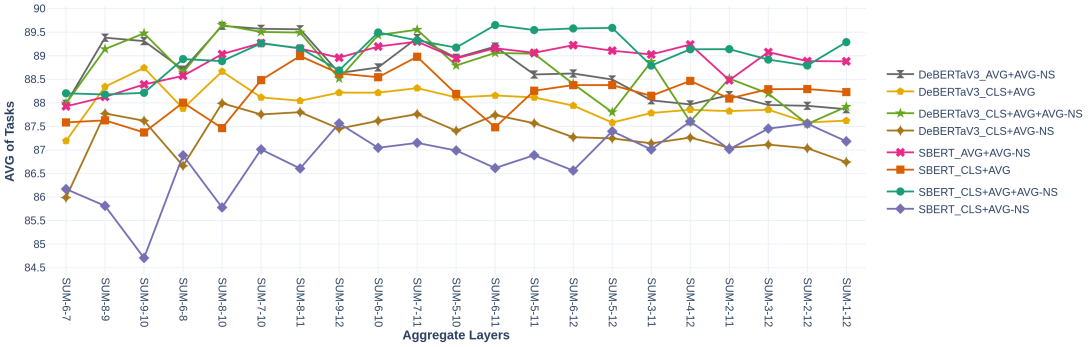



Figure 4: Average task accuracy for concatenating two and three pooling vectors. Results for DeBERTa and SBERT, with the best result for each size of layer aggregations for both two and three vectors presented.

89.43. In contrast, SBERT's best two-token configuration reached only 89.41, slightly below the single-token result.

The layer aggregation trends observed for concatenation align with those in Figure 3, with top-performing single-token aggregations also excelling in multi-token settings. These findings suggest that combining multiple pooling methods is a promising strategy for enhancing document representations.

### 4.6 Comparison with baselines

To compare the impact of pooling and layer aggregation relative to existing embedding methods, Table 1 presents the results of previously proposed methods. It also presents our best pooling and layer aggregation strategies for 1, 2, and 3 tokens when considering the **validation set** average task accuracy. We used the validation set for choosing the configurations to ensure a fair comparison with the baselines, as these choises function as hyperparameters.

As shown in Table 1, careful selection of pooling and aggregation strategies significantly improves performance, even with a single-token representation. Our best result—DeBERTa with concatenated `[CLS]`, AVG, and AVG-NS pooled from the sum of layers 8–11 achieved an average accuracy of 89.49, surpassing the strongest baseline, suSupSimCSE-RB-LC (Hosseini et al., 2023), which reached 88.66 after 3 epochs of training.

To isolate the effect of pooling and aggregation, we compare original encoder results with their optimized single-token configurations. SBERT's average accuracy increased from 85.27 to 89.42 by summing layers 5–12 and applying AVG pooling, a nearly 5% improvement. DeBERTa showed even greater gains, from 81.51 to 88.88, marking a 9% increase. These improvements underscore the substantial, often overlooked, impact of pooling and

| Model | MR | CR | SUBJ | MPQA | SST2 | TREC | MRPC | Average |
|---|---|---|---|---|---|---|---|---|
| RoBERTa$_{\text{AVG/Last Layer}}$ | 80.46 | 72.24 | 93.01 | 80.60 | 87.48 | 74.80 | 66.49 | 79.30 |
| DeBERTa$_{\text{AVG/Last Layer}}$ | 83.71 | 73.40 | 92.46 | 83.01 | 90.55 | 78.20 | 69.22 | 81.51 |
| BERT$_{\text{AVG/Last Layer}}$ ⋆ | 78.66 | 86.25 | 94.37 | 88.66 | 84.40 | 92.80 | 69.45 | 84.94 |
| SBERT Δ | 80.10 | 86.25 | 94.61 | 88.78 | 84.90 | 89.00 | 73.25 | 85.27 |
| su-HNCSE-PM ⊙ [1ep] | 81.94 | 86.99 | 95.20 | 89.77 | 86.81 | 85.31 | 75.49 | 85.93 |
| su-AoE-BERTΔ [1ep] | 83.00 | 89.38 | 94.72 | 89.87 | 87.20 | 89.00 | 75.54 | 86.96 |
| su-CLTC-RoBERTa† [5ep] | 86.86 | **92.35** | 94.31 | 89.91 | 91.36 | 87.73 | 77.14 | 88.52 |
| su-SRoBERTa-B-LC♢ [200iter] | 85.76 | 91.75 | 94.80 | 90.51 | 90.78 | 87.95 | **78.92** | 88.64 |
| su-SimCSE-RoBERTa-MLM△ [3ep] | 85.08 | 91.76 | 94.02 | 89.72 | 92.31 | 91.20 | 76.52 | 88.66 |
| SBERT$_{\text{AVG / SUM-5-12}}$ | 85.93 | 90.36 | 95.30 | 90.56 | 91.65 | **96.00** | 76.23 | 89.43 |
| DeBERTa$_{\text{AVG / SUM-7-10}}$ | 86.50 | 88.77 | 95.10 | 90.27 | 92.42 | 93.80 | 75.30 | 88.88 |
| SBERT$_{\text{AVG+AVG-NS / SUM-8-12}}$ | 85.99 | 89.72 | 94.81 | 90.42 | 90.94 | 95.20 | 77.04 | 89.16 |
| DeBERTa$_{\text{AVG+AVG-NS / SUM-6-10}}$ | 86.89 | 90.23 | 95.67 | 90.73 | 92.15 | 94.40 | 71.19 | 88.75 |
| SBERT$_{\text{CLS+AVG+AVG-NS / SUM-7-12}}$ | 86.19 | 91.02 | **95.77** | **90.76** | 91.93 | **96.00** | 74.90 | 89.48 |
| DeBERTa$_{\text{CLS+AVG+AVG-NS / SUM-8-11}}$ | **87.47** | 90.12 | 95.39 | 90.43 | **92.59** | 95.20 | 75.25 | **89.49** |

Table 1: Results of sentence embeddings on classification tasks for the baselines and the best 1,2, and three token alternatives for SBERT and DEBERTA on the validation set. The reported metric is accuracy. ⋆: (Reimers and Gurevych, 2019); Δ: (Li and Li, 2024); ⊙: (Liu et al., 2024b); †: (Liu et al., 2024a); △: (Gao et al., 2021); ♢: (Hosseini et al., 2023).

aggregation choices in sentence representation.

On individual tasks, our best pooling and aggregation strategies, combined with token concatenation, achieved top results on five of the seven tasks (MR, SUBJ, MPQA, SST2, and TREC). In MRPC, while not leading, our methods outperformed the vanilla encoders (BERT, DeBERTa, SBERT, and RoBERTa with last-layer AVG pooling). Across the full range of configurations tested, our results remained consistently strong, demonstrating that pooling and aggregation are critical factors in optimizing transformer-based embeddings for classification tasks.

## 5 Conclusion

In this work, we systematically studied the impact of different pooling strategies and layer aggregation techniques in Transformer encoder-only models for text classification. Our analysis covered widely used pooling methods, including `[CLS]`, mean, sum, and max pooling, along with novel combinations of these techniques. We also examined the effects of extracting embeddings from different model layers and proposed aggregation strategies to enhance sentence representations. Additionally, we introduced the concatenation of multiple pooling vectors as a way to further improve performance.

Our experimental results show that pooling selection and layer aggregation significantly affect text classification accuracy. By carefully choosing the pooling method and combining multiple layers, we achieved improvements of up to 9% over traditional approaches using the same encoder. Notably, our study show that standard practices, such as relying solely on the [CLS] token, using only the last layer's embeddings, or averaging tokens based on a single layer, often yield suboptimal results. Instead, combining multiple pooling techniques and leveraging intermediate layers leads to more robust representations, especially when summing layers and averaging tokens. We believe our findings may provide a roadmap for optimizing Transformer encoders in text classification tasks.

These results highlight the untapped potential in Transformer-encoded sentence and document representations, opening several venues for future work. A key next step is developing a method to automatically select pooling and aggregation parameters, better adapting to the encoder's characteristics and the task at hand. We are actively working on this automatic selection process.

Furthermore, we believe that integrating layer aggregation and pooling into encoder training could further enhance classification performance, and we plan to explore strategies for retraining existing encoders with optimized pooling/aggregation configurations. Finally, we aim to extend our analysis to large Transformer encoders (24 layers) and similarity tasks, as well as investigate additional aggregation techniques, such as weighted averages and concatenating poolings from non-consecutive layers.

## Limitations

While we believe this work is an important first step in demonstrating the impact of pooling, layer aggregation, and pooling concatenation in classification tasks, there are several scenarios and limitations that could not be addressed within the scope of a single paper. In this section, we highlight what we consider the most important aspects not tackled in this work.

- **A systemized approach to automatically choose pooling and layer aggregations**. In this paper, we are interested in showing the full potential of pooling and layer aggregation that is mostly not being used by Transformer encoder-based methods. We did not focus, however, in **how** to automatically select the best configurations.While we offer general insights and recurring patterns across encoders, the optimal settings remain model-specific, as can be seen in the fact that in SBERT using both the last and deeper layers (5 and 6) lead to the best results overall, unlike DeBERTa. We believe that defining heuristics and, more importantly, developing automatic methods to select pooling strategies is essential to fully realize the potential demonstrated here. We are currently investigating such approaches.

- **Comparison with autoregressive LLM models**. Our baselines focus on Transformer encoder-based models, as our contributions, pooling, aggregation, and token concatenation, are designed to improve this class of models. We exclude autoregressive LLMs such as (Li and Li, 2024) from the main comparison due to their fundamentally different architecture (e.g., using prompts to compress information into a single token) and significantly larger scale (7–13 billion parameters versus 110–140 million in BERT-base models). Nonetheless, we include a comparison in Appendix B.

- **Similarity tasks**. Our primary focus is on classification tasks, while much of the existing literature emphasizes similarity tasks for both training and evaluation. We recognize the relevance of similarity tasks, particularly in applications like RAG, and have conducted experiments in this setting using our pooling and aggregation strategies. However, we chose to center our analysis on classification to provide a clearer directional evaluation. Results for similarity tasks using our best classification configurations are presented in Appendix A.

- **More sophisticated layer aggregations**. In this paper, we explored summing and averaging consecutive layers. While this lead to good results, obviously more sophisticated approaches, such as weighting different layers, choosing non consecutive layers, and using machine learning to determine the aggregation are possible and must be further evaluated.

## Acknowledgements

Supressed for double blind review

## References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Yuho Cha and Younghoon Lee. 2024. Advanced sentence-embedding method considering token importance based on explainable artificial intelligence and text summarization model. *Neurocomputing*, 564:126987.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Harsh Dubey, Yulu Qin, and Rahul Meghwal. 2023. Layer by layer-examining bert's syntactic and semantic representation. *XXX*.

Yuchen Fu, Zifeng Cheng, Zhiwei Jiang, Zhonghui Wang, Yafeng Yin, Zhengliang Li, and Qing Gu. 2024. Token prepending: A training-free approach for eliciting better sentence embeddings from llms. *arXiv preprint arXiv:2412.11556*.

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

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

CAO Hongliu. 2024. Recent advances in text embedding: A comprehensive review of top-performing methods on the mteb benchmark. *Google Scholar*.

MohammadSaleh Hosseini, Munawara Munia, and Latifur Khan. 2023. Bert has more to offer: Bert layers combination yields better sentence embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15419–15431.

Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, Fan Yang, Mengnan Du, and Yongfeng Zhang. 2024. Exploring concept depth: How large language models acquire knowledge at different layers? *Preprint*, arXiv:2404.07066.

Xianming Li and Jing Li. 2024. Aoe: Angle-optimized embeddings for semantic textual similarity. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1825–1839.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2022. A survey of transformers. *AI open*, 3:111–132.

Chenjing Liu, Xiangru Chen, Peng Hu, Jie Lin, Junfeng Wang, and Xue Geng. 2024a. Contrastive learning with transformer initialization and clustering prior for text representation. *Applied Soft Computing*, 166:112162.

Wenxiao Liu, Zihong Yang, Chaozhuo Li, Zijin Hong, Jianfeng Ma, Zhiquan Liu, Litian Zhang, and Feiran Huang. 2024b. Hncse: Advancing sentence embeddings via hybrid contrastive learning with hard negatives. *arXiv preprint arXiv:2411.12156*.

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.

Zhu Liu, Cunliang Kong, Ying Liu, and Maosong Sun. 2024c. Fantastic semantics and where to find them: Investigating which layers of generative llms reflect lexical semantics. *Preprint*, arXiv:2403.01509.

Rajvardhan Patil, Sorio Boit, Venkat Gudivada, and Jagadeesh Nandigam. 2023. A survey of text representation and embedding techniques in nlp. *IEEE Access*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. *OpenAI*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Lukas Stankevičius and Mantas Lukoševičius. 2024. Extracting sentence embeddings from pretrained transformer models. *Applied Sciences*, 14(19):8887.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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

Jintang Xue, Yun-Cheng Wang, Chengwei Wei, and C-C Jay Kuo. 2024. Efficient feature selection for word embedding dimension reduction. *x*.

## A Similarity Tasks

While the main focus of this work is on classification, we also evaluate our pooling and layer aggregation strategies on semantic textual similarity (STS) tasks, which are widely adopted for benchmarking sentence embeddings. An interesting aspect to consider when exploring these new pooling and aggregation methods is how they affect similarity tasks. Thus, we present the results of our best pooling and layer aggregation strategies on the classification validation set on semantic textual similarity (STS) tasks, which are widely adopted for benchmarking sentence embeddings. We use the standard seven STS datasets from the SentEval suite: STS12–STS16, STS-Benchmark (STS-B), and SICK-Relatedness (SICK-R). These benchmarks compute the Spearman correlation between the cosine similarity of sentence embeddings and human-labeled similarity scores.

Table 2 presents the results for baseline models and our best-performing configurations. As expected, SBERT significantly outperforms vanilla transformer models (e.g., BERT and RoBERTa) when used with average pooling on the last layer. This is due to the fact that SBERT was specifically trained to capture semantic similarity relationships between sentences, using a siamese archi-

| Model | STS12 | STS13 | STS14 | STS15 | STS16 | STS-B | SICK-R | Average |
|---|---|---|---|---|---|---|---|---|
| RoBERTa$_{\text{AVG/Last Layer}}$ | 39.47 | 17.49 | 24.41 | 31.96 | 45.55 | 69.61 | 76.80 | 43.61 |
| DeBERTa$_{\text{AVG/Last Layer}}$ | 38.08 | 27.46 | 34.52 | 38.82 | 36.90 | 48.74 | 66.06 | 41.51 |
| BERT$_{\text{AVG/Last Layer}}$ ⋆ | 38.78 | 57.98 | 57.98 | 63.15 | 61.06 | 46.35 | 58.40 | 54.81 |
| SBERT Δ | 70.97 | 76.53 | 73.19 | 79.09 | 74.30 | 77.03 | 72.91 | 74.89 |
| su-AoE-BERT Δ | 75.26 | 85.61 | 80.64 | 86.36 | 82.51 | 85.64 | 80.99 | 82.43 |
| su-HNCSE-PM ⊙ | 71.02 | 83.92 | 75.52 | 82.93 | 81.03 | 81.45 | 72.76 | 78.38 |
| su-CLTC-RoBERTa † | **77.85** | **88.96** | **81.72** | **86.63** | **83.98** | 84.27 | 82.12 | **83.65** |
| su-SimCSE-RoBERTa-MLM △ | 76.53 | 85.21 | 80.95 | 86.03 | 82.57 | **85.83** | 80.50 | 82.52 |
| su-SRoBERTa-B-LC ♢ | 72.94 | 76.14 | 72.83 | 82.29 | 77.13 | 78.99 | 76.90 | 76.75 |
| SBERT$_{\text{AVG / SUM-5-12}}$ | 68.04 | 68.85 | 70.64 | 80.25 | 80.10 | 77.08 | 81.65 | 75.23 |
| DeBERTa$_{\text{AVG / SUM-7-10}}$ | 52.64 | 47.21 | 54.93 | 67.67 | 63.38 | 62.00 | 75.09 | 60.42 |
| SBERT$_{\text{AVG+AVG-NS / SUM-8-12}}$ | 69.46 | 72.38 | 72.43 | 81.46 | 81.02 | 78.18 | 82.44 | 76.77 |
| DeBERTa$_{\text{AVG+AVG-NS / SUM-6-10}}$ | 54.57 | 51.28 | 57.16 | 69.37 | 64.09 | 54.87 | 65.51 | 59.55 |
| SBERT$_{\text{CLS+AVG+AVG-NS / SUM-7-12}}$ | 66.88 | 68.86 | 69.66 | 78.03 | 80.22 | 65.91 | **82.45** | 73.14 |
| DeBERTa$_{\text{CLS+AVG+AVG-NS / SUM-8-11}}$ | 51.67 | 43.32 | 53.31 | 65.73 | 60.82 | 60.51 | 75.38 | 58.68 |

Table 2: Spearman's correlation scores across seven STS benchmarks for various models. ⋆: (Reimers and Gurevych, 2019); Δ: (Li and Li, 2024); ⊙: (Liu et al., 2024b); †: (Liu et al., 2024a); △: (Gao et al., 2021); ♢: (Hosseini et al., 2023). The remaining lines are the best pooling and layer aggregation configurations in the classification validation set.

tecture and supervised objectives aimed at bringing semantically equivalent sentences closer together. This focus naturally makes it more suitable for STS benchmarks, where the evaluation metric is precisely the semantic closeness between sentence pairs. Among baselines, SimCSE and CLTC-RoBERTa deliver the highest correlation scores across all tasks, demonstrating the effectiveness of contrastive learning for similarity.

Our pooling and aggregation methods, despite being developed without supervision or fine-tuning, show competitive results. The best-performing configuration for similarity was the combination AVG+AVG-NS applied over the sum of layers 8 to 12 of SBERT, which achieved an average Spearman correlation of 76.77, surpassing the original SBERT baseline (74.89). This indicates that carefully selecting intermediate layers and combining pooling strategies can yield improvements even in tasks the models were not directly optimized for.

Interestingly, while our configurations show solid results with SBERT, DeBERTa's performance lags behind on similarity tasks despite excelling in classification. For instance, DeBERTa with AVG+AVG-NS over layers 6–10 reached only 59.55 on average. This suggests that while pooling and aggregation improve classification, similarity tasks may benefit more from supervised fine-tuning or contrastive training, particularly for encoders like DeBERTa that were not originally designed for sentence-level semantics.

These findings reinforce our hypothesis: pooling and aggregation are key factors in sentence representation, but optimal configurations may differ between classification and similarity tasks. Future work will further explore strategies that unify both objectives under a shared representation framework. Also, in future work, it will be important to perform fine-tuning of the encoders themselves to further optimize their performance for similarity tasks and potentially align them better with pooling and aggregation strategies.

## B Comparison with decoder-only LLMs

We also present a comparison our best results with recent decoder-only large language models (LLMs) to contextualize their relative performance and resource requirements.

Table 3 presents the classification accuracy of several LLM-based methods, such AoE-LLaMA and PromptEOL, alongside our best pooling and aggregation configurations for SBERT and DeBERTa. Notably, LLM-based approaches typically rely on prompt-based encoding, where the input sentence is embedded into a fixed position within a prompt and the final token embedding is extracted to represent the full sentence. While effective, these methods depend heavily on prompt engineering and incur substantial computational costs, often involving models with 7 to 13 billion parameters or even much larger.

Despite this, our encoder-only methods, particularly those using DeBERTa (86M parameters) with optimized pooling and aggregation (e.g., CLS+AVG+AVG-NS over SUM-8-11), achieved an average accuracy of 89,49, 1.93 smaller than

| Model | MR | CR | SUBJ | MPQA | SST2 | TREC | MRPC | Average |
|---|---|---|---|---|---|---|---|---|
| AoE-Llama7B | 90.54 | 93.06 | 96.14 | 91.61 | 95.00 | 95.80 | 74.90 | 91.01 |
| AoE-Llama13B ♢ | 90.77 | 93.01 | 96.15 | **91.83** | 94.95 | 96.60 | 76.87 | **91.45** |
| PromptEOL (Lllama2-7B) | 90.63 | 92.87 | 96.32 | 91.19 | 95.00 | 95.40 | 75.19 | 90.94 |
| PromptEOL + TP (Ours) (Lllama2-7B) △ | **90.90** | **93.35** | **96.58** | 91.51 | 95.50 | 96.00 | 76.12 | 91.42 |
| Pretended CoT (Lllama2-7B) | 90.10 | 92.24 | 96.32 | 91.54 | 95.11 | 94.20 | 75.77 | 90.75 |
| Pretended CoT + TP (Ours) (Lllama2-7B) | 90.45 | 92.61 | 96.52 | 91.59 | **95.77** | 96.00 | 76.81 | 91.39 |
| Knowledge (Lllama2-7B) | 89.84 | 93.03 | 96.21 | 91.54 | 94.78 | 97.20 | 73.91 | 90.93 |
| Knowledge + TP (Ours) (Lllama2-7B) | 90.39 | 93.32 | 96.31 | 91.56 | 94.51 | **97.60** | 76.06 | 91.39 |
| $\text{SBERT}_{\text{AVG / SUM-5-12}}$ | 85.93 | 90.36 | 95.30 | 90.56 | 91.65 | 96.00 | 76.23 | 89.43 |
| $\text{DeBERTa}_{\text{AVG / SUM-7-10}}$ | 86.50 | 88.77 | 95.10 | 90.27 | 92.42 | 93.80 | 75.30 | 88.88 |
| $\text{SBERT}_{\text{AVG+AVG-NS / SUM-8-12}}$ | 85.99 | 89.72 | 94.81 | 90.42 | 90.94 | 95.20 | 77.04 | 89.16 |
| $\text{DeBERTa}_{\text{AVG+AVG-NS / SUM-6-10}}$ | 86.89 | 90.23 | 95.67 | 90.73 | 92.15 | 94.40 | 71.19 | 88.75 |
| $\text{SBERT}_{\text{CLS+AVG+AVG-NS / SUM-7-12}}$ | 86.19 | 91.02 | 95.77 | 90.76 | 91.93 | 96.00 | 74.90 | 89.48 |
| $\text{DeBERTa}_{\text{CLS+AVG+AVG-NS / SUM-8-11}}$ | 87.47 | 90.12 | 95.39 | 90.43 | 92.59 | 95.20 | 75.25 | 89.49 |
| ⋆ DeBERTa 86M versus Llama 7B △ | -3.42 | -3.23 | -1.19 | -1.08 | -2.91 | -0.8 | -0.87 | -1.93 |
| ⋆ DeBERTa 86M versus Llama 13B ♢ | -3,3 | -2,89 | -0.76 | -6.4 | -2.36 | -1.4 | -1.62 | -1.96 |

Table 3: Results of sentence embeddings on classification tasks for decoder-only LLM baselines and the best 1,2, and three token alternatives for SBERT and DeBERTa on the validation set. The reported metrics is accuracy. ♢: (Li and Li, 2024); △: (Fu et al., 2024).

AoE-LLaMA-13B (91.45), while using only 0.66% of its parameters, and PromptEOL+TP (91.42), with 1.23% of its parameters, offering a vastly more efficient alternative in resource-constrained scenarios.

These results highlight two key insights: Pooling and aggregation significantly close the performance gap between encoder-only models and much larger LLMs in sentence encoding for classification tasks; and Large language models do not inherently outperform optimized encoders unless fine-tuned extensively or used with well-crafted prompts and templates. While decoder-only LLMs are increasingly adopted across NLP tasks, our findings demonstrate that enhanced sentence embeddings from smaller, frozen encoders, leveraging careful pooling and layer aggregation, can remain competitive.