Words That Matter: Vocabulary Pruning in ModernBERT

Wout De Rijck

Supervisors: Prof. Dr. Ir. Thomas Demeester, Prof. Dr. Ir. Chris Develder
Councellor: Dr. Ir. Karel D'Oosterlinck

Abstract: Large language models (LLMs) require substantial computational and memory resources, limiting their utility in resource-constrained environments. ModernBERT is a smaller encoder-only language model that excels at various tasks while being computationally highly efficient. In this work, we study how much more ModernBERT can be compressed while retaining accuracy on any given task. Specifically, we introduce a series of very simple vocabulary pruning techniques that target the embedding layer. We compare the resulting accuracy with LoSparse, a state-of-the-art gradient-based pruning technique that targets the encoder layers. For parameter reductions up to ∼20%, our much simpler vocabulary pruning technique outperforms LoSparse, retaining up to 97.6% of ModernBERT's performance across various tasks, compared to LoSparse's 96.2%. The strong performance of our simple technique indicates that task-specific pruning can meaningfully increase the efficiency of ModernBERT, an already highly efficient model. Additionally, our results suggest that state-of-the-art encoder-layer pruning can fall short of simple embedding-layer pruning. We provide open-source implementations of all pruning methods and evaluation tools to support further research in this area. (`https://github.com/WoutDeRijck/vocab-pruning.git`)

Keywords: vocabulary pruning, ModernBERT, encoder-only model, model compression, embedding-layer, task-specific pruning, TF-IDF

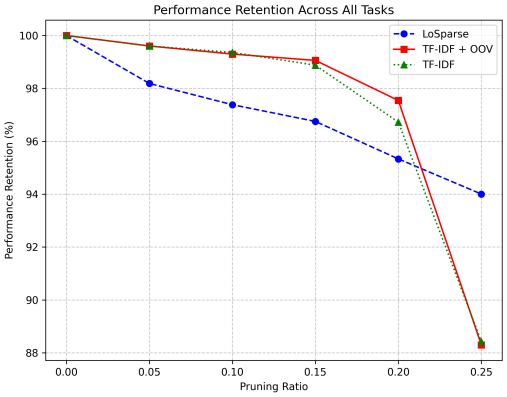

Figure 1: Simple vocabulary pruning techniques retain >97% of ModernBERT's performance while removing 20% of total parameters across eight GLUE benchmark tasks. Our vocabulary pruning removes tokens based on their importance using *TF-IDF*, with Out-of-Vocabulary (OOV) handling that maps removed tokens to semantically similar retained ones, substantially outperforming LoSparse—a state-of-the-art gradient-based encoder pruning method. Pruning ratio represents the fraction of total parameters removed from the model, limited to 0.25 since the embedding layer constitutes 25.95% of all model parameters, representing the theoretical maximum for vocabulary-only pruning.

1 Introduction

Language models need to be efficient for optimal deployment. Efficiency can be assessed in several dimensions, including inference latency, throughput, energy consumption, parameter count, and memory usage [1]. In this work, we study how to optimally reduce the memory footprint of language models by removing unimportant vocabulary tokens from the embedding layer, resulting in faster model loading, reduced deployment costs, and improved accessibility for edge devices or resource-constrained environments.

ModernBERT [2] stands out as one of the most lightweight and optimized encoder-only transformers available. It integrates several architectural and training innovations while maintaining robust downstream performance. Despite these advancements, we investigate whether ModernBERT's efficiency can be further enhanced through parameter pruning. Given ModernBERT's inherent efficiency, optimal pruning strategies could yield exceptionally efficient models.

Specifically, we explore task-specific weight pruning, as ModernBERT is intended to be finetuned on specific tasks for optimal behavior (unlike more general-purpose decoder-only models that can utilize in-context learning). We hypothesize that simple task-specific pruning of the embedding matrix can be highly efficient, as many tokens can be irrelevant for any specific downstream task, and the embedding layers counts for a significant portion (∼25%) of the total parameters [3].

We explore a series of simple vocabulary pruning heuristics on a range of Natural Language Processing tasks, including linguistic acceptability, sentiment analysis, paraphrase detection, semantic similarity, question answering, and natural language inference, as cov-

ered by the GLUE benchmark [4]. For any task, we first apply vocabulary pruning to the base ModernBERT model and subsequently finetune the pruned model. We compare these vocabulary pruning techniques with LoSparse [5], a state-of-the-art gradient-based pruning technique that targets the encoder layers.

Our results show that TFIDF-based vocabulary pruning (with frequencies computed from the target task's training dataset) outperforms encoder-layer pruning (LoSparse) for compression ratios up to 20%, maintaining 97.6% of ModernBERT's original performance while removing 77.34% of embedding parameters (see Figure 1). This shows that substantial efficiency gains can be achieved with virtually no performance loss using a simple, offline pruning technique. Beyond 20% compression, performance drops sharply as we approach the embedding layer's parameter limit (25.95% of total model parameters), while LoSparse maintains more stable performance at higher ratios, since it can continue pruning across the larger encoder layers (74.05% of total parameters). Unlike LoSparse, which requires computationally expensive gradient-based optimization, our vocabulary pruning operates as a pre-finetuning step with minimal overhead.

Our results indicate that the memory footprint of light-weight models, such as ModernBERT, can still be significantly improved. We open-source all our code.

2 Related work

Model compression techniques enhance the efficiency of large language models (LLMs) by reducing their size and computational requirements. These techniques can be broadly categorized into four primary methods [6–10]: (i) **quantization**, which reduces the numerical precision of model weights [11]; (ii) **parameter pruning**, which eliminates redundant or less important connections and neurons to create sparser models with fewer parameters [12]; (iii) **low-rank approximation**, which decomposes weight matrices into lower-dimensional representations that capture essential information while requiring fewer parameters [13]; and (iv) **knowledge distillation** transfers knowledge from larger teacher models to smaller student models, enabling competitive performance with reduced architecture size [14].

The model compression techniques described above are complementary and can be combined to achieve optimal compression [15–17]. This work focuses on parameter pruning techniques to reduce model size while preserving performance. The rest of this section gives an overview of the most important parameter pruning techniques.

Encoder-Layer Pruning methods remove less critical parameters within the encoder to reduce model size while maintaining acceptable accuracy. A representative state-of-the-art method is LoSparse [5], which decomposes each weight matrix into a low-rank component and a sparse residual. During finetuning, LoSparse calculates the importance of each weight by multiplying it by its gradient, which is resource-intensive.

Attention-head pruning [18] removes entire attention heads, whereas LoSparse removes entire neurons (i.e., rows in weight matrices). Movement pruning [19] applies gradient-guided masking to gradually zero out low-importance weights during finetuning. LayerDrop [20] introduces structured dropout, allowing entire transformer layers to be stochastically removed during training and optionally pruned at inference.

These encoder-layer pruning methods differ in their granularity and strategy—targeting individual weights, attention heads, or full layers. LoSparse stands out as a strong baseline for encoder-level pruning [21–23].

Embedding-Layer Pruning methods reduce the memory footprint of the embedding layer, which can comprise up to 20-30% of a model's parameters. These methods can be categorized into two main strategies: embedding compression and vocabulary pruning.

Embedding compression reduces the dimensional representation of the embeddings. LightToken [24] is a model- and task-agnostic framework for compressing the token embedding layers. It uses a combination of low-rank approximation, a residual binary autoencoder, and a novel compression loss to drastically reduce embedding size.

Vocabulary Pruning Techniques are a specialized form of embedding pruning that removes rarely used or task-irrelevant tokens.

The simplest approach is to drop unused tokens. TextPruner [25] scans a corpus and removes tokenizer tokens not present in the target text. This can reduce $\text{BERT}_{\text{BASE}}$'s embedding matrix by up to 47% without compromising performance.

For multilingual models, most of the parameters are located in the embedding layer, language-specific pruning has proven effective. Abdaoui et al. [26] showed that trimming unused language tokens from mBERT can achieve a 45% parameter reduction with comparable accuracy on XNLI. Similarly, BLADE [27] builds pruned bilingual models containing only tokens from target languages, reducing parameters by $\sim$36.5% versus full mBERT while speeding up training by $\sim$30% and inference by $\sim$55%.

While simply removing embeddings for unseen (i.e., never-used) tokens has proven to be effective, we show one can go substantially further by additionally ranking and pruning low-utility tokens.

To our knowledge, this is the first study to (i) rigorously evaluate a range of importance heuristics in a single framework, (ii) incorporate OOV-aware mapping to preserve semantic fidelity of pruned tokens, and (iii) apply vocabulary pruning atop ModernBERT thereby demonstrating the practical benefits of embedding-layer reduction in resource-constrained settings.

3 Task-Specific Vocabulary Pruning

Our vocabulary pruning method is straightforward: we first remove all tokens that do not appear in our training dataset, then we eliminate an additional percentage of the remaining tokens based on importance scores. Optionally, we cluster removed tokens into groups and add the cluster centroids back into the vocabulary to handle out-of-vocabulary (OOV) cases during inference.

We apply this method to ModernBERT, where the embedding layer accounts for approximately 25% of the model's parameters. This approach is based on the observation that tokens in a vocabulary have varying importance for downstream tasks, and that certain tokens can be selectively removed with minimal impact on performance.

In contrast to methods that require resource-intensive post-pruning finetuning, the proposed vocabulary pruning approach operates as a **pre-finetuning offline pruning** step in the model adaptation pipeline. It requires only the training dataset to determine token importance and prune the vocabulary, without needing gradients from finetuning, allowing the resulting pruned model to be finetuned with minimal computational overhead.

We compute token importance scores using either attention-based (Algorithm 1, lines 2-4) or statistical approaches (line 6) via `compute_importance_scores`, then sort and select the top k tokens based on the pruning ratio p (lines 7-9). The least important tokens are pruned from the embedding layer, and the embedding matrix is reconstructed using only the retained tokens (`rebuild_model_with_vocab`, line 13), yielding a smaller model. To recover the loss of throwing away tokens, semantic clustering can be applied with `cluster_pruned_tokens` to map pruned tokens to semantically similar retained tokens (lines 10-12). This approach front-loads the pruning work to the pre-finetuning phase, meaning the actual finetuning process remains unchanged and operates on an already-reduced model. With a pruning ratio of 0, our algorithm defaults to standard finetuning of the base model.

Algorithm 1 Pre-Finetuning Vocabulary Pruning

Require: Model M with vocab V, dataset D, pruning ratio p
Ensure: Pruned model M' with reduced vocabulary
1: $V_s \leftarrow \text{get\_special\_tokens}(M)$ ▷ [CLS], [SEP]
2: **if** using attention scoring **then**
3: $M_{\text{ft}} \leftarrow \text{finetune}(M, D)$ ▷ Finetune model on task
4: $s \leftarrow \text{compute\_attention\_scores}(M_{\text{ft}}, D, V)$
5: **else**
6: $s \leftarrow \text{compute\_token\_scores}(D, V)$
7: **end if**
8: $V_{\text{sorted}} \leftarrow \text{sort}(V \setminus V_s, s, \text{desc})$
9: $k \leftarrow \lfloor (1-p) \cdot |V_{\text{sorted}}| \rfloor$ ▷ Number of tokens to keep
10: $V_{\text{keep}} \leftarrow V_s \cup V_{\text{sorted}}[1:k]$
11: **if** using OOV handling **then**
12: $\text{clusters} \leftarrow \text{cluster\_pruned\_tokens}(V \setminus V_{\text{keep}})$
13: $\text{oov\_map} \leftarrow \text{map\_pruned\_to\_representatives}(\text{clusters})$
14: **end if**
15: $M' \leftarrow \text{rebuild\_model\_with\_vocab}(M, V_{\text{keep}}, \text{oov\_map})$
16: **return** M'

3.1 Token Importance Analysis

A critical challenge in vocabulary pruning is determining which tokens to retain and which to remove. We explore several token importance estimation techniques, ranging from simple statistical methods to semantic analysis.

Random Selection prunes tokens randomly without consideration for importance, serving as a baseline approach. In this method, tokens are selected for removal using uniform random sampling from the vocabulary.

Clustering-Based preserves semantic diversity across the vocabulary using DBSCAN [28] clustering to group similar token embeddings and keeping one representative per cluster. This leverages the property that embeddings with high cosine similarity typically have similar meanings [29], ensuring maximal semantic coverage with reduced vocabulary size.

Frequency-Based ranks tokens by their frequency in the training dataset, pruning the least frequently used tokens first. This approach is grounded in the heuristic that infrequent tokens contribute less to model performance [30].

We explore two frequency-based ranking methods. *Simple Frequency* counts raw token occurrences in the training data, favoring common tokens. *TF-IDF* (Term Frequency-Inverse Document Frequency) balances token occurrence with discriminative power across documents. This method prioritizes tokens that appear frequently in specific documents but are rare across the corpus, capturing task-specific terminology while filtering out ubiquitous tokens that carry less semantic value.

Attention-Based leverages the attention mechanism of transformer models to identify potentially important tokens. The underlying principle is that tokens receiving higher attention during inference might be those the model relies on more when making predictions, offering a model-based perspective on token importance [31–34]. While attention scores do not necessarily imply causal relevance, they provide a task-specific signal that differs from simple statistical measures. The approach first finetunes the base model on the target task to learn task-specific attention patterns, then processes the dataset through this model to capture attention matrices from all layers and heads. It aggregates the attention each token receives across all its occurrences, and finally normalizes scores by token frequency to avoid bias toward common tokens.

3.2 Out-of-Vocabulary Token Handling

Vocabulary pruning raises an important question: how should the model process tokens that were removed from its vocabulary? Handling these out-of-vocabulary (OOV) tokens can recover performance lost due to pruning. One option is to map all pruned tokens to a single UNK token, discarding their semantic information. Alternatively, semantic clustering can be used to maintain some of the original meaning of pruned tokens.

	Single Sentence		Paraphrase and Similarity			Natural Language Inference			
Method	**SST-2**	**CoLA**	**MRPC**	**STS-B**	**QQP**	**MNLI**	**QNLI**	**RTE**	**AVG**
Baseline									
ModernBERT (Full)	0.951	0.632	0.89	0.917	0.917	0.881	0.939	0.643	0.846
Vocabulary Pruning									
Train Tokens Only	0.950	0.630	0.861	0.917	0.917	0.883	0.915	0.639	0.839
Parameter Reduction	*18.85%*	*22.61%*	*18.56%*	*18.76%*	*4.79%*	*6.74%*	*6.42%*	*17.06%*	*14.22%*
Random Selection	0.911	0.470	0.798	0.845	0.780	0.504	0.669	0.566	0.693
Clustering-Based	0.894	0.188	0.752	0.696	0.871	0.683	0.833	0.566	0.685
Attention-Based	**0.947**	0.589	**0.864**	0.885	0.763	0.683	0.791	0.578	0.763
Simple Frequency-Based	0.943	0.467	0.812	0.905	0.904	0.858	0.902	0.546	0.792
TF-IDF Based	0.933	0.520	0.807	0.901	0.898	0.860	0.909	0.610	0.805
Simple Frequency + OOV	0.938	0.540	0.828	**0.906**	**0.915**	0.858	0.907	0.615	0.813
TF-IDF + OOV	0.934	**0.615**	0.834	0.903	0.912	0.858	**0.910**	**0.635**	**0.825**
Parameter Reduction	*20.25%*	*23.27%*	*20.02%*	*20.18%*	*20.02%*	*20.02%*	*20.02%*	*20.02%*	*20.48%*
Encoder Layer Pruning									
LoSparse	0.935	0.525	0.856	0.887	**0.915**	**0.871**	0.906	0.614	0.814
Parameter Reduction	*20.16%*	*20.16%*	*20.16%*	*20.16%*	*20.16%*	*20.16%*	*20.16%*	*20.16%*	*20.16%*

Table 1: Performance on GLUE dev set. ModernBERT is fine-tuned separately for each task. Scores are accuracies except for CoLA (Matthew's correlation), and STS-B (Pearson correlation). Notation "+ OOV" indicates pruning with out-of-vocabulary clustering. At 20% parameter reduction, TF-IDF + OOV maintains 97.6% of the original performance and outperforms LoSparse (0.825 vs 0.810 average score). OOV handling improves results by 2.0 percentage points on average.

The clustering-based Out-Of-Vocabulary (OOV) handling process extracts embeddings for all pruned tokens and applies K-means clustering to group semantically similar tokens together. From each cluster, the token closest to the centroid is selected as a representative. While alternative clustering methods (e.g., DBSCAN) and representative selection strategies (e.g., averaging embeddings) could be explored, we focus on this approach for its simplicity and effectiveness. During inference, when an OOV token is encountered, it is mapped to its assigned representative. The number of clusters (k) offers a tunable parameter to balance compression rate against semantic precision. While this approach adds a few more tokens back into the vocabulary, the intuition is that it can recover most of the performance lost during pruning.

After evaluating the different token importance metrics, we apply clustering-based OOV handling to the best-performing approach to further improve the results.

4 Experiments

We evaluate our vocabulary pruning methods across multiple natural language understanding tasks to assess their effectiveness. Additionally, we compare them to baseline approaches.

For datasets and metrics, The General Language Understanding Evaluation (GLUE) benchmark [4] serves as our primary evaluation framework, comprising eight diverse NLP tasks: single-sentence tasks (SST-2 [35], CoLA [36]), similarity/paraphrase tasks (MRPC [37], STS-B [38], QQP [39]), and inference tasks (MNLI [40], QNLI [41], RTE [42]). We report task-appropriate metrics: accuracy for most classification tasks, Matthew's correlation for CoLA, and Pearson correlation for STS-B.

As baselines, we establish three baselines to evaluate our vocabulary pruning techniques: (i) *ModernBERT (Full)* serves as the upper performance bound, using the complete 50,368-token vocabulary and all encoder parameters, finetuned on the training dataset of each task; (ii) *LoSparse* provides an encoder-layer pruning baseline that uses low-rank and sparse factorization to preserve 80% of parameters in encoder layers while maintaining the full embedding layer; (iii) *Train Tokens Only* represents a simple vocabulary reduction approach that removes tokens not observed in finetuning data, achieving variable parameter reduction (4.79-22.61%) depending on the dataset's vocabulary coverage.

The results in Table 1 present a comprehensive evaluation across GLUE tasks for different pruning methods. Vocabulary pruning achieves substantial parameter reduction (20.02-23.27%) with minimal performance impact, with *TF-IDF + OOV* maintaining 97.6% of the original model's performance while reducing parameters by over 20%. OOV handling mechanisms boost results compared to their non-OOV counterparts (+2.0 percentage points on average).

The substantial performance degradation observed with random selection (0.693 average vs 0.846 baseline) demonstrates that strategic token selection is crucial for vocabulary pruning effectiveness.

Vocabulary pruning outperforms encoder-layer pruning (LoSparse) at lower pruning ratios (5-20%), while only using simple statistics before finetuning, as shown in Figure 2. Specifically, a 20% reduction in total parameters (77.34% of vocabulary size) maintains competitive performance.

The efficacy of vocabulary pruning can be explained by the token-distribution: 20% of tokens account for 80-94% of all occurrences in training sets across GLUE

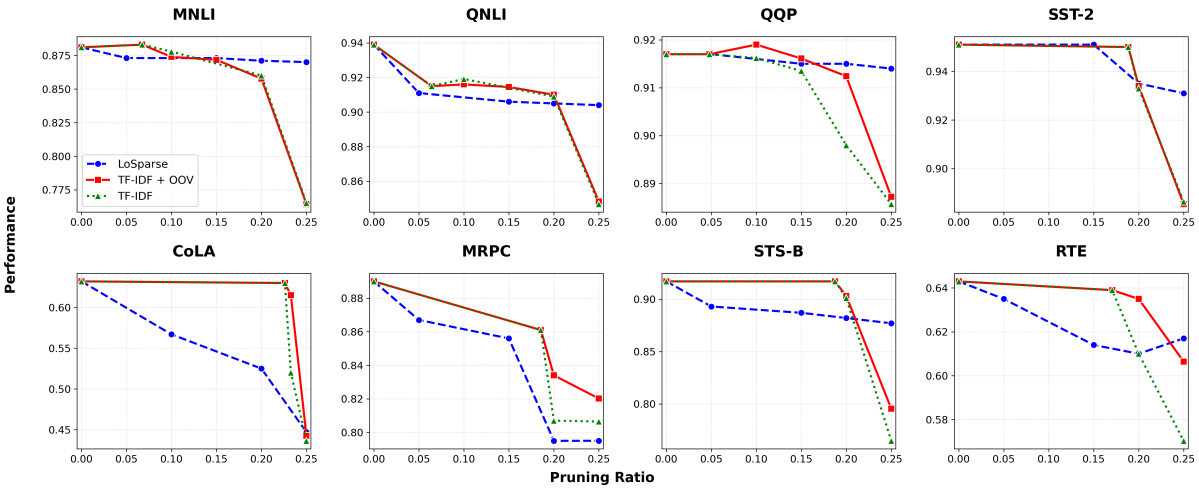

Figure 2: Performance comparison between vocabulary pruning (*TF-IDF* with and without OOV) and encoder pruning (LoSparse). Vocabulary pruning outperforms LoSparse at lower pruning ratios (5-20%), but performance drops sharply beyond 20% as the embedding layer's parameter limit is reached, while LoSparse maintains more stable performance at higher compression rates.

tasks, as shown in Table 2. This redundancy provides strong empirical justification for our approach.

The table also highlights varying train-test vocabulary overlap across tasks, with unseen token percentages ranging from 0.14% to 13.70%. These high rates of unseen tokens for some tasks explain why effective OOV handling improves performance by providing semantic mappings for unseen tokens rather than discarding their meaning.

The embedding layer's 25.95% parameter share imposes an upper limit on vocabulary pruning, with performance declining beyond the 20% threshold while LoSparse maintains more stable performance at higher compression rates.

To assess the practical efficiency improvements, we benchmark the computational and memory requirements of our vocabulary pruning approach. The benchmark results in Table 3 demonstrate that our approach delivers 20% storage reduction and 14.83% lower GPU memory use with negligible inference time impact (+1.33%).

Task	Unique Tokens		Vocab Cov. (%)		Top 20%		OOV
	Train	Test	Train	Test	Train	Test	%
COLA	5,416	1,934	17.74	6.34	85.12	74.68	8.74
MNLI	25,793	22,664	84.51	74.25	90.69	89.64	0.14
MRPC	11,096	3,567	36.35	11.69	83.17	71.80	13.04
QNLI	26,176	20,360	85.76	66.71	88.31	85.79	0.41
QQP	25,486	18,534	83.50	60.72	94.02	91.79	1.03
RTE	13,354	4,198	43.75	13.75	81.77	70.75	12.12
SST2	11,536	8,084	37.80	26.49	84.23	80.53	0.42
STSB	10,346	3,271	33.90	10.72	82.66	71.30	13.70

Table 2: Token statistics across GLUE tasks. Vocab Cov.: percentage of model vocabulary present in data. Top 20%: token coverage by most frequent 20% of vocabulary. OOV%: test tokens not in training.

Metric	Base	Pruned	Impr.(%)
Params (M)	149.61	119.69	20.00
Embed. Params (M)	38.68	8.76	77.34
Storage (MB)	570.72	456.59	20.00
GPU Mem (MB)	823.06	700.99	14.83
Infer. Time (ms)	18.73	18.98	-1.33

Table 3: Benchmark results comparing base ModernBERT with pruned variant (TF-IDF + OOV) during inference on an RTX 4090. Our method achieves 20% reduction in total parameters (77.34% of embedding parameters) and a 14.83% reduction in GPU memory with negligible impact on inference time.

5 Conclusion

We find that vocabulary pruning strategies can outperform more advanced encoder-layer pruning methods, for pruning ratios up to 20% on the ModernBERT encoder model. Our *TFIDF-based* method with OOV handling achieved 97.6% of the original model's performance across GLUE tasks while reducing total parameters by 20.02%, compared to LoSparse's 96.2% retention at the same compression ratio. This resulted in practical benefits of 14.83% reduction in GPU memory requirements and 20% less storage space.

These results challenge the typical prioritization of encoder-layer pruning over vocabulary reduction and highlight the significant redundancy within embedding layers of pre-trained models, providing an efficient path to model compression for resource-constrained environments.

6 Future Work and Limitations

While our methods show promise, several limitations remain. Our clustering-based method for OOV handling relies on K-means clustering, while alternative methods such as DBSCAN [28] might better capture complex semantic relationships in embedding spaces without requiring pre-specified cluster numbers. Similarly, our attention-based token importance method relies solely on attention scores, but recent work suggests this may be incomplete, as investigation into value vector norms reveals notably non-uniform patterns [31]. Future work should explore Value-Aware Token Pruning (VATP) approaches that combine attention scores with the ℓ_1 norm of value vectors for more comprehensive token importance evaluation.

Additionally, this work focuses exclusively on encoder-only models where embedding layers constitute approximately 25% of parameters. In contrast, decoder-only models and encoder-decoder architectures typically have much larger transformer blocks relative to their embedding layers, meaning vocabulary pruning would yield substantially smaller compression benefits as embeddings represent only a small fraction of total parameters. This embedding layer constraint fundamentally limits our vocabulary pruning to 25% total parameter reduction, beyond which performance degrades sharply. However, future research could explore combining vocabulary pruning with encoder-layer compression techniques like LoSparse to achieve higher compression ratios, applying LoSparse to the remaining 75% of encoder parameters on top of optimal vocabulary pruning to potentially exceed either method's individual performance.

Finally, our approach is inherently task-specific, requiring separate pruning and finetuning for each target task, which increases deployment complexity compared to general-purpose compression methods, though it enables more aggressive compression aligned with ModernBERT's intended use case.

References

[1] Zhengqing Yuan, Weixiang Sun, et al. Efficientllm: Efficiency in large language models. *arXiv preprint arXiv:2505.13840*, 2025.

[2] Benjamin Warner, Antoine Chaffin, et al. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference, 2024.

[3] Nakyeong Yang et al. Task-specific compression for multi-task language models using attribution-based pruning. *arXiv preprint arXiv:2205.04157*, 2022.

[4] Alex Wang, Amanpreet Singh, et al. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[5] Zixuan Li, Meiqi Ding, et al. Losparse: Structured pruning via low-rank and sparse decomposition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

[6] Zhongwei Wan, Xin Wang, et al. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863*, 2023. Accepted to Transactions on Machine Learning Research (TMLR), May 2024.

[7] Seungcheol Park et al. A comprehensive survey of compression algorithms for language models. *arXiv preprint arXiv:2401.15347*, 2024.

[8] Yu Cheng et al. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[9] Rahul Mishra et al. A survey on deep neural network compression: Challenges, overview, and solutions. *arXiv preprint arXiv:2010.03954*, 2020.

[10] Xailient. 4 popular model compression techniques explained. `https://xailient.com/blog/4-popular-model-compression-techniques-explained/`, 2021.

[11] Amir Gholami et al. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[12] Hongrong Cheng et al. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *arXiv preprint arXiv:2308.06767*, 2023.

[13] Edward J. Hu, Yelong Shen, et al. Lora: Low-rank adaptation of large language models, 2021.

[14] Geoffrey Hinton, Oriol Vinyals, et al. Distilling the knowledge in a neural network, 2015. Technical report.

[15] Saurav Muralidharan et al. Compact language models via pruning and knowledge distillation. *arXiv preprint arXiv:2407.14679*, 2024.

[16] Song Han et al. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[17] Yoni Choukroun et al. Low-bit quantization of neural networks for efficient inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2019.

[18] Paul Michel, Omer Levy, et al. Are sixteen heads really better than one? In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[19] Victor Sanh, Albert Webson Xu, et al. Movement pruning: Adaptive sparsity by fine-tuning. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[20] Angela Fan, Edouard Grave, et al. Reducing transformer depth on demand with structured dropout. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[21] Guangyan Li, Yongqiang Tang, et al. Lorap: Transformer sub-layers deserve differentiated structured compression for large language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 28657–28672, 23–29 Jul 2024.

[22] Owen Huang. Awesome llm compression. `https://github.com/HuangOwen/Awesome-LLM-Compression`, 2023. Accessed: 2025-05-19.

[23] Yujia Ren and Yifan Zhu. Large language model compression with global rank and sparsity optimization. *arXiv preprint arXiv:2505.03801*, 2025.

[24] Yifei Wang et al. LightToken: Structured embedding compression for transformer inference. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[25] Yelong Shen et al. TextPruner: A unified framework for pruning token-level and feature-level redundancy in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[26] Amine Abdaoui, Arnaud Fehrentz, et al. Load what you need: BERT optimized for speed. *arXiv preprint arXiv:2007.02450*, 2020.

[27] Suraj Nair, Jean Maillard, et al. BLADE: Combining vocabulary pruning and intermediate pretraining for scaleable neural clir. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2023.

[28] Wang and Daren others. Dbscan: Optimal rates for density-based cluster estimation. *Journal of Machine Learning Research*, 20(170):1–50, 2019.

[29] Tomas Mikolov, Kai Chen, et al. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

[30] Yuhao Li, Tao Xia, et al. Enhancing performance of transformer-based models in natural language understanding through word importance embedding. *Knowledge-Based Systems*, 287:111443, 2024.

[31] Zhiyu Guo, Hidetaka Kamigaito, et al. Attention score is not all you need for token importance indicator in kv cache reduction: Value also matters. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 21158–21166, November 2024.

[32] Sehoon Kim et al. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 784–794, 2022.

[33] Saurabh Goyal et al. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 3690–3699, 2020.

[34] Xiaohan Chen et al. Earlybert: Efficient bert training via early-bird lottery tickets. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 2195–2207, 2021.

[35] Richard Socher et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642, 2013.

[36] Alex Warstadt et al. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics (TACL)*, 7:625–641, 2019.

[37] William B Dolan et al. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[38] Daniel Cer et al. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, 2017.

[39] Quora question pairs. `https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs`. Accessed: 2025-06-01.

[40] Adina Williams et al. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1112–1122, 2018.

[41] Pranav Rajpurkar et al. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392, 2016.

[42] Ido Dagan et al. The pascal recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, 2005.