

# Words That Matter: Vocabulary Pruning in ModernBERT

Wout De Rijck

Supervisors: Prof. Dr. Ir. Thomas Demeester, Prof. Dr. Ir. Chris Develder

Councillor: 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  $\sim 20\%$ , our much simpler vocabulary pruning technique outperforms LoSparse, retaining up to 97.6% of ModernBERT’s performance across various tasks, compared to LoSparse’s 92.9%. 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.<sup>12</sup>

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

## 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. 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 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

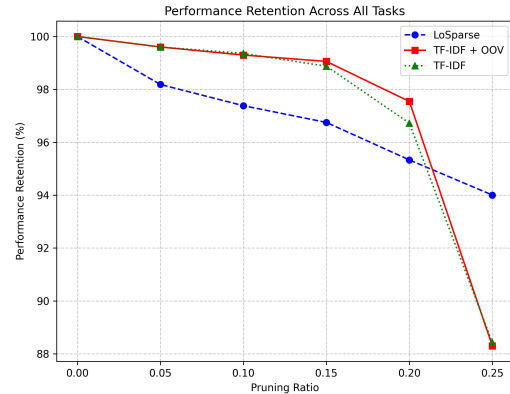


Figure 1: Average performance retention across all GLUE tasks. Our vocabulary pruning removes tokens based on their importance using TF-IDF, with Out-of-Vocabulary (OOV) handling that maps removed tokens to semantically similar retained ones. LoSparse is a gradient-based pruning technique that targets encoder layers through low-rank plus sparse factorization. Pruning ratio represents the fraction of total parameters removed from the model, limited to 0.25 as the embedding layer constitutes 25.95% of all parameters.

performance. Despite these advancements, a critical question remains open: can we further enhance ModernBERT’s efficiency by pruning parameters?

This question is particularly interesting for two reasons. First, ModernBERT’s inherent efficiency means that optimal pruning strategies could yield exceptionally efficient models. Second, existing research into pruning techniques have predominantly targeted decoder-only LLMs [13], leaving encoder-only models relatively underexplored.

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 ( $\sim 25\%$  of the total parameters).

We explore a series of simple vocabulary pruning heuristics on a range of Natural Language Pro-

<sup>1</sup><https://github.com/WoutDeRijck/vocab-pruning.git>

<sup>2</sup><https://github.com/WoutDeRijck/LoSparse.git>

cessing tasks including linguistic acceptability, sentiment analysis, paraphrase detection, semantic similarity, question answering, and natural language inference, as covered by the GLUE benchmark. 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, a state-of-the-art gradient-based pruning technique that targets the encoder layers.

Our results show that TF-IDF-based vocabulary pruning 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. Unlike LoSparse, which requires computationally expensive gradient-based optimization, our vocabulary pruning operates as a straightforward 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 and pruned models.

## 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 [17]: (i) **quantization**, which reduces the numerical precision of model weights; (ii) **parameter pruning**, which eliminates redundant or less important connections and neurons to create sparser models with fewer parameters; (iii) **low-rank approximation**, which decomposes weight matrices into lower-dimensional representations that capture essential information while requiring fewer parameters [5]; and (iv) **knowledge distillation** transfers knowledge from larger teacher models to smaller student models, enabling competitive performance with reduced architecture size. [4]

The model compression techniques described above are complementary and can be combined to achieve optimal compression. This work focuses on parameter pruning techniques to reduce model size while preserving performance. The rest of this section is dedicated to enumerating 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 [9], which de-

composes 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 [10] removes entire attention heads, whereas LoSparse removes entire neurons (i.e., rows in weight matrices). Movement pruning [15] applies gradient-guided masking to gradually zero out low-importance weights during finetuning. Layer-Drop [2] 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. [7] [6] [14]

**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 [19] 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 [16] scans a corpus and removes tokenizer tokens not present in the target text. This can reduce BERT<sub>BASE</sub>’s embedding matrix by up to 47% without compromising performance.

For multilingual models, most of the parameters are located in the embeddings layer, language-specific pruning has proven effective. Abdaoui et al. [1] showed that trimming unused language tokens from mBERT can achieve a 45% parameter reduction with comparable accuracy on XNLI. Similarly, BLADE [12] 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—a state-of-the-art lightweight encoder—thereby demonstrating the practical benefits of embedding-layer reduction in resource-constrained settings.

## Method

We propose a task-specific vocabulary pruning method for ModernBERT that targets the embedding layer, which 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. We explore and compare several token importance estimation techniques, ranging from simple statistical methods to semantic analysis. The method consists of three components: token importance analysis, selective pruning, and an optional semantic clustering for out-of-vocabulary (OOV) tokens that can further enhance performance in some cases.

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. Instead, it relies on either simple statistical measures or a single attention analysis pass, which does not require gradient calculations, resulting in minimal computational overhead before standard finetuning.

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 via `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:  $V_s \leftarrow \text{get\_special\_tokens}(M)$   $\triangleright$  [CLS], [SEP]
2: if using attention scoring then
3:    $M_{ft} \leftarrow \text{finetune}(M, D)$   $\triangleright$  Finetune model on task
4:    $s \leftarrow \text{compute\_attention\_scores}(M_{ft}, D, V)$ 
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:  $k \leftarrow \lfloor (1 - p) \cdot |V_{\text{sorted}}| \rfloor$   $\triangleright$  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  $M'$ 
```

---

## Token Importance Analysis

A critical challenge in vocabulary pruning is determining which tokens to retain and which to remove. Our approach first retains only tokens that appear in the training dataset, removing those never encountered during training. From this reduced set, we then apply more targeted pruning techniques to further decrease vocabulary size.

**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 by grouping similar tokens together and keeping only one representative from each group. It leverages the property that embeddings pointing in similar directions (high cosine similarity) typically have similar meanings [11]. This process ensures maximal semantic coverage with a 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. [8]

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 [3]. 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.

## 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

Method	Single Sentence		Paraphrase and Similarity			Natural Language Inference			AVG
	SST-2	CoLA	MRPC	STS-B	QQP	MNLI	QNLI	RTE	
<b>Baseline</b>									
ModernBERT (Full)	0.951	0.632	0.89	0.917	0.917	0.881	0.939	0.643	0.846
<b>Vocabulary Pruning</b>									
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	0.899	0.051	0.714	0.786	0.774	0.501	0.510	0.566	0.600
Attention-Based	<b>0.947</b>	0.589	<b>0.864</b>	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	<b>0.860</b>	<b>0.909</b>	0.610	0.805
Simple Frequency + OOV	0.938	0.540	0.828	<b>0.906</b>	<b>0.915</b>	0.858	0.907	0.615	0.813
TF-IDF + OOV	0.923	<b>0.615</b>	0.834	0.903	0.912	0.858	0.908	<b>0.635</b>	<b>0.824</b>
Parameter Reduction	20.25%	23.27%	20.02%	20.18%	20.02%	20.02%	20.02%	20.02%	20.48%
<b>Encoder Layer Pruning</b>									
LoSparse	0.929	0.525	0.856	0.882	0.911	0.858	0.907	0.610	0.810
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 original performance and outperforms LoSparse (0.824 vs 0.810 average score). OOV handling improves results by 1.9 percentage points on average.

UNK token, discarding their semantic information. Alternatively, semantic clustering can be used to maintain some of the original meaning of pruned tokens.

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. 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.

## Experiments

We evaluate our vocabulary pruning methods across multiple natural language understanding tasks to assess the effectiveness of our vocabulary pruning methods and compare them to baseline approaches.

**For datasets and metrics,** The General Language Understanding Evaluation (GLUE) benchmark [18] serves as our primary evaluation framework, comprising eight diverse NLP tasks: single-sentence tasks (SST-2, CoLA), similarity/paraphrase tasks (MRPC, STS-B, QQP), and inference tasks (MNLI, QNLI, RTE). 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 baseline models

to evaluate our vocabulary pruning techniques: (i) *ModernBERT (Full)* serves as the upper performance bound, employing the complete 50,368-token vocabulary and all encoder parameters, finetuned on each task; (ii) *LoSparse* provides a complementary 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 dataset vocabulary coverage. These baselines establish benchmarks for assessing both parameter efficiency and performance of our proposed methods.

**For results,** Table 1 presents 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 (+1.9 percentage points on average).

Pruning effectiveness varies by task: attention-based methods excel on single-sentence tasks and paraphrase detection, while TFIDF-based methods perform better on complex reasoning tasks. We hypothesize this is because simpler tasks rely on specific sentiment-related tokens that receive high attention during inference, while complex reasoning requires a broader set of discriminative tokens that TF-IDF effectively identifies across documents. As shown in Figure 2, vocabulary pruning outperforms encoder-layer pruning (LoSparse) at lower pruning ratios (5-20%), while only using simple statistics before finetuning. A 20% reduction in to-

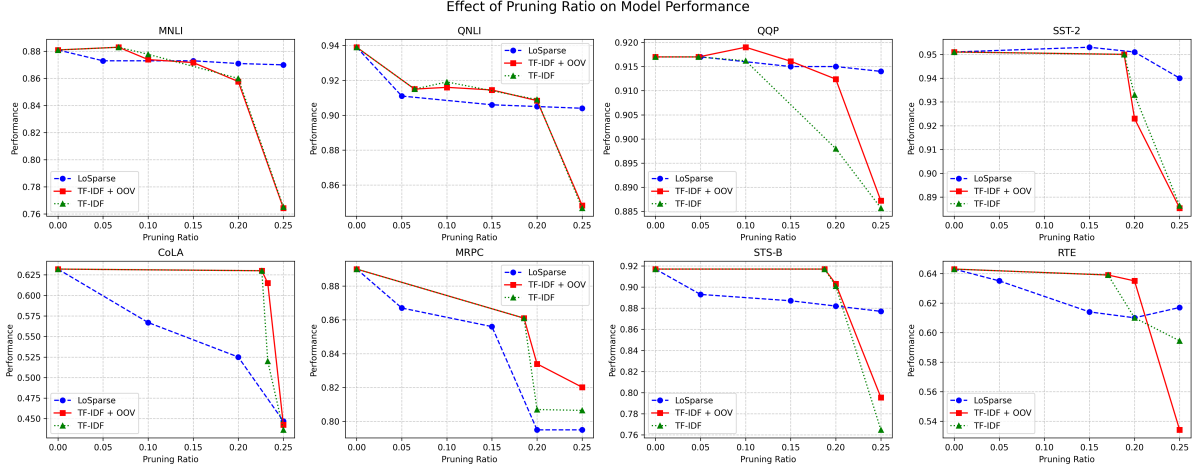


Figure 2: Performance comparison between vocabulary pruning (TF-IDF + OOV) and encoder pruning (LoSparse). Vocabulary pruning outperforms LoSparse at lower pruning ratios (5-20%), but performance drops sharply beyond 20% as the embedding layer’s parameter limit is reached, while LoSparse maintains more stable performance at higher compression rates.

tal parameters (77.34% of vocabulary size) maintains competitive performance, making this approach particularly valuable for resource-constrained environments. Vocabulary pruning outperforms encoder-layer pruning (LoSparse) at lower pruning ratios (5-20%), while only using simple statistics before fine-tuning, as shown in Figure 2.

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 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 OOV percentages ranging from 0.14% to 13.70%. These high OOV rates 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.

Practically, 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%).

## 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 TF-IDF-based method with OOV handling achieved 97.6% of the original model’s performance across GLUE tasks while reducing total pa-

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.

rameters by 20.02%, compared to LoSparse’s 92.9% 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.

## Future Work

Future research could explore an integrated approach combining vocabulary pruning with encoder-layer compression techniques like LoSparse. Since vocabulary pruning cannot exceed 25% of total parameters (the embedding layer’s size), and our results in Figure 1 show vocabulary pruning outperforms LoSparse up to 20%, we hypothesize that applying LoSparse on top of 20% vocabulary pruning would yield better results than using LoSparse alone at 25% compression.

## References

- [1] Amine Abdaoui, Arnaud Fehrentz, and Thomas Lavergne. Load what you need: BERT optimized for speed. *arXiv preprint arXiv:2007.02450*, 2020.
- [2] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *ICLR*, 2020.
- [3] Zhiyu Guo, Hidetaka Kamigaito, and Taro Watanabe. 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*, pages 21158–21166, November 2024.
- [4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. Technical report.
- [5] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.
- [6] Owen Huang. Awesome llm compression. <https://github.com/HuangOwen/Awesome-LLM-Compression>, 2023. Accessed: 2025-05-19.
- [7] Guangyan Li, Yongqiang Tang, and Wensheng Zhang. Lorap: Transformer sub-layers deserve differentiated structured compression for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 28657–28672, 23–29 Jul 2024.
- [8] Yuhao Li, Tao Xia, Jiahao Pan, Wei Wu, and Tao Liu. Enhancing performance of transformer-based models in natural language understanding through word importance embedding. *Knowledge-Based Systems*, 287:111443, 2024.
- [9] Zixuan Li, Meiqi Ding, Zhou Yu, Tuo Zhao, and Tengyu Ma. Lospase: Structured pruning via low-rank and sparse decomposition. In *International Conference on Learning Representations (ICLR)*, 2023.
- [10] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *NeurIPS*, 2019.
- [11] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.
- [12] Suraj Nair, Jean Maillard, Carsten Eickhoff, James Mackenzie, and J. Shane Culpepper. BLADE: Bilingual lexicon-aware document expansion for multilingual dense retrieval. In *Proceedings of the 46th International ACM SIGIR Conference*, 2023.
- [13] Raghuvver Namburi, Nadav Rotem, Minjoon Kwon, Leonid Izhikevich, Lisa Chai, Ritchie Tao, Alex Vance, Sam Shleifer, Cliff Wang, et al. LLM in a flash: Efficient large language model inference with limited memory. *arXiv preprint arXiv:2309.16104*, 2023.
- [14] Yujia Ren and Yifan Zhu. Large language model compression with global rank and sparsity optimization. *arXiv preprint arXiv:2505.03801*, 2025.
- [15] Victor Sanh, Albert Webson Xu, Colin Raffel, Sam Shleifer, Stephen Liu, Ajit Subramani, and Alexander M Rush. Movement pruning: Adaptive sparsity by fine-tuning. In *NeurIPS*, 2020.
- [16] Yelong Shen et al. TextPruner: A unified framework for pruning token-level and feature-level redundancy in text. In *EMNLP*, 2022.
- [17] Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, and Mi Zhang. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863*, 2023. Accepted to Transactions on Machine Learning Research (TMLR), May 2024.
- [18] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.
- [19] Yifei Wang et al. LightToken: Structured embedding compression for transformer inference. In *NeurIPS*, 2023.