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 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.

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.

From this perspective, ModernBERT stands out as one of the most lightweight and optimized encoder-only transformers available. It integrates several architectural and training innovations, successfully maintaining robust downstream performance while minimizing inference costs. Despite these advancements, a critical question remains open: can we further enhance ModernBERT's efficiency by pruning parameters?

This question is particularly significant for two reasons. First, ModernBERT's inherent efficiency and its strong potential for large-scale deployment mean that optimal pruning strategies on top of this model could yield exceptionally efficient models. Second, existing research into pruning techniques have predominantly targeted decoder-only LLMs [7] due to their widespread popularity, leaving encoder-only models relatively underexplored.

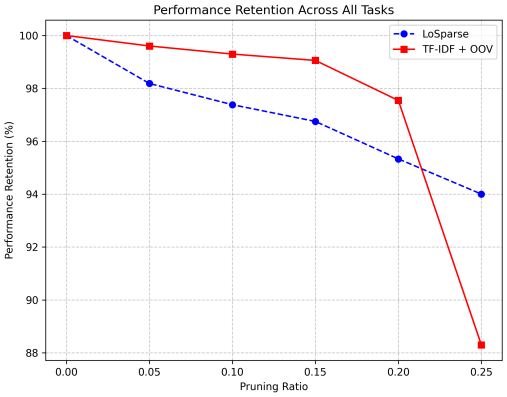

Figure 1: Average Performance Retention Across All Tasks for different pruning methods. Pruning ratio represents the fraction of total parameters removed from the model, limited to 0.25 as the embedding layer constitutes 25.95% of all parameters.

Specifically, we explore task-specific weight pruning, as ModernBERT is intended to be fine-tuned 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).

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 covered by the GLUE benchmark. For any task, we first apply this pruning to the base ModernBERT model and subsequently fine-tune the pruned model. For any given pruning ratio, we measure the accuracy retention compared

to simply fine-tuning the model. We compare these vocabulary pruning techniques with LoSparse, a state-of-the-art gradient-based pruning technique that targets the encoder layers.

As shown in Figure 1, 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. 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-fine-tuning step with minimal overhead, making it particularly suitable for resource-constrained environments requiring moderate model compression.

Our results indicate that the memory footprint of light-weight models, such as ModernBERT, can still be significantly improved. Additionally, the superior performance of our simple vocabulary pruning methods highlights the need to further develop pruning methods. 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: (i) **quantization**, which reduces the numerical precision of model weights, decreasing memory footprint; (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; and (iv) **knowledge distillation**, which transfers knowledge from larger teacher models to smaller student models, enabling competitive performance with reduced architecture size [10].

These approaches are complementary and can be combined to achieve optimal compression. This work focuses on structured 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 transformer encoder to reduce model size while maintaining acceptable accuracy. A representative state-of-the-art method is LoSparse [4], which decomposes each weight matrix into a low-rank component and a sparse residual. During fine-tuning, LoSparse calculates the importance of each weight by multiplying it with its gradient.

Attention-head pruning [5] removes entire attention heads. Movement pruning [8] applies gradient-guided masking to gradually zero out low-importance weights during fine-tuning. LayerDrop [3] 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. Among them, LoSparse stands out as a strong baseline for encoder-level pruning.

Embedding-Layer Pruning methods target the embedding layer, which can comprise up to 20-30% of a model's parameters. These approaches can be broadly categorized into two main strategies: (1) vocabulary pruning, which removes entire token entries, and (2) embedding compression, which retains all tokens but reduces their dimensional representation.

Embedding compression reduces the memory footprint of the embedding matrix through dimensionality reduction techniques. LightToken [12] 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 remove rarely used or task-irrelevant tokens to shrink the model's vocabulary. These methods simply delete rows in the embedding matrix, making them straightforward to implement.

The simplest approach is "drop unused tokens." Both TextPruner [9] and Partial Embedding Matrix Adaptation (PEMA; [2]) scan a corpus and remove tokenizer tokens not present in the target text. TextPruner permanently deletes these entries, while PEMA temporarily removes them during fine-tuning and restores them afterward. These methods can reduce $\text{BERT}_{\text{BASE}}$'s embedding matrix by up to 47% without compromising performance.

For multilingual models, 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 [6] 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%.

These findings consistently demonstrate that vocabulary pruning is an effective way to tailor models to specific domains or languages with minimal performance impact.

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. The method primarily consists of two main components: token importance analysis and selective pruning, with an optional third component: semantic clustering for out-of-vocabulary (OOV) tokens that can further enhance performance in some cases.

Pre-Fine-Tuning Pruning Procedure

In contrast to methods that require resource-intensive post-pruning fine-tuning, the proposed vocabulary pruning approach operates as a pre-fine-tuning offline optimization step in the model adaptation pipeline.

Compared to the standard workflow for adapting ModernBERT to a downstream task, which involves taking the pre-trained base model and fine-tuning it directly on the task data, we first analyze the task-specific dataset to extract token statistics. Tokens are then ranked based on their importance and the least important tokens are pruned from the embedding layer.

The pruning techniques follow the same procedure, formalized in Algorithm 1. The embedding matrix is then reconstructed using only the retained tokens, yielding a smaller model. Optional semantic clustering can be applied to map pruned tokens to semantically similar retained tokens,

This approach front-loads the pruning work to the pre-fine-tuning phase, meaning the actual fine-tuning process remains unchanged and operates on a model that already has a reduced parameter count. The resulting pruned model maintains its standard inference pipeline while benefiting from reduced memory requirements.

Algorithm 1 Pre-Fine-Tuning Vocabulary Pruning

Require: Pretrained model M with vocabulary V, downstream dataset D, pruning ratio p
Ensure: Pruned model M'

stats $\leftarrow$ compute_token_statistics(D)
scores $\leftarrow$ rank_tokens(stats)
$V_{\text{special}} \leftarrow$ special_tokens(M) ▷ ([CLS], [SEP], etc.)
$V_{\text{keep}} \leftarrow V_{\text{special}} \cup$ top_k_tokens$(\text{scores}, 1-p)$
$M' \leftarrow$ rebuild_model_with_vocab(M, V_{keep})
return M'

Token Importance Analysis

A critical challenge in vocabulary pruning is determining which tokens to retain and which to remove. This research explores and compares several token importance estimation techniques, each with distinct approaches and practical implications for model performance. These approaches range from simple statistical methods to complex 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 full vocabulary.

Clustering-Based preserves semantic diversity across the vocabulary while reducing redundancy 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.

The approach uses agglomerative hierarchical clustering on normalized embedding vectors to: (i) *group* semantically similar tokens into clusters, (ii) *identify* the token closest to each cluster's centroid as its representative, and (iii) *retain* only these representatives while pruning all other tokens. This process ensures maximal semantic coverage with a reduced vocabulary size.

Frequency-Based ranks tokens by their frequency in the target dataset, with least frequently used tokens pruned first. This approach operates on the heuristic that rarely used tokens contribute less to model performance.

We explore two frequency-based ranking methods:

Simple Frequency counts raw token occurrences in the training data, favoring common tokens regardless of their distribution across documents or examples.

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.

The TF-IDF approach operates by: (i) *calculating* term frequency (TF) as the normalized count of a token in each document, (ii) *computing* inverse document frequency (IDF) as the logarithm of the ratio between total documents and documents containing the token, (iii) *multiplying* these values to get TF-IDF scores that highlight discriminative tokens, and (iv) *normalizing* scores (using L2 normalization in our experiments) to prevent outlier tokens from dominating.

Task	Total Tokens		Unique Tokens		Vocab Coverage (%)		Top 20% Coverage (%)		Test Vocabulary Coverage	
	Train	Test	Train	Test	Train	Test	Train	Test	OOV Tokens	OOV%
COLA	71,818	7,939	5,416	1,934	17.74	6.34	85.12	74.68	169	8.74%
MNLI	13,044,457	1,452,279	25,793	22,664	84.51	74.25	90.69	89.64	31	0.14%
MRPC	165,585	18,701	11,096	3,567	36.35	11.69	83.17	71.80	465	13.04%
QNLI	4,388,343	485,747	26,176	20,360	85.76	66.71	88.31	85.79	83	0.41%
QQP	9,036,384	1,000,855	25,486	18,534	83.50	60.72	94.02	91.79	191	1.03%
RTE	151,109	16,215	13,354	4,198	43.75	13.75	81.77	70.75	509	12.12%
SST2	686,566	75,775	11,536	8,084	37.80	26.49	84.23	80.53	34	0.42%
STSB	128,151	14,481	10,346	3,271	33.90	10.72	82.66	71.30	448	13.70%

Table 1: Token statistics across GLUE tasks, comparing train and test splits. OOV Tokens: number of unique tokens in the test set that do not appear in the training set; OOV%: percentage of test vocabulary not found in train.

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. While attention scores don't necessarily imply causal relevance, they provide a task-specific signal that differs from simple statistical measures. Attention-based pruning aims to preserve tokens that appear to participate more actively in the model's decision-making process.

The approach operates by: (i) *fine-tuning* the base model on the target task to learn task-specific attention patterns, (ii) *processing* the dataset through this model to capture attention matrices from all layers and heads, (iii) *aggregating* the attention each token receives across all its occurrences, and (iv) *normalizing* scores by token frequency to avoid bias toward common tokens.

Out-of-Vocabulary Token Handling

Another question is what happens with tokens that are removed from the vocabulary. One option is to map them to a single UNK token, effectively discarding their semantic information. Alternatively, semantic clustering can be used to maintain some of the original meaning of pruned tokens.

The clustering-based 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 rather than the generic UNK token. The number of clusters (k) offers a tunable parameter to balance compression rate against semantic precision.

We will extend the frequency-based approaches with this clustering-based OOV handling.

Experiments

After developing several algorithmic approaches to vocabulary pruning, this section presents a comprehensive evaluation designed to answer several key questions; can selective vocabulary reduction maintain model performance while significantly decreasing parameter count? How do different pruning strategies compare on various natural language understanding tasks? Does the pre-fine-tuning approach that avoids post-pruning recovery fine-tuning deliver practical benefits in real-world deployment scenarios? Through careful experimentation across diverse tasks, the results demonstrate not only the effectiveness of these algorithms but also their practical advantages in creating more efficient transformer models.

Datasets and Metrics The General Language Understanding Evaluation (GLUE) benchmark [11] 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.

Table 1 reveals significant vocabulary redundancy across GLUE tasks, with the top 20% of tokens covering 80-94% of all occurrences in training sets—providing strong empirical justification for vocabulary pruning. The table also examines train-test vocabulary overlap, which varies significantly across tasks and underscores the importance of effective OOV handling strategies in the pruning process.

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; (ii) *LoSparse* provides a complementary encoder-layer pruning baseline that uses low-rank plus 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 embeddings not observed in fine-tuning 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.

	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
Encoder Layer Pruning									
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%*	
Vocabulary Pruning									
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
Frequency-Based	0.943	0.467	0.812	0.905	0.904	0.858	0.902	0.546	0.792
Attention-Based	**0.947**	0.589	**0.864**	0.885	0.763	0.683	0.791	0.578	0.763
TF-IDF Based	0.933	0.520	0.807	0.901	0.898	**0.860**	**0.909**	0.574	0.800
Freq + OOV	0.938	0.540	0.828	**0.906**	**0.915**	0.858	0.907	0.615	0.813
TF-IDF + OOV	0.923	**0.615**	0.834	0.903	0.912	0.858	0.908	**0.635**	**0.824**
Parameter Reduction	*20.25%*	*23.27%*	*20.02%*	*20.18%*	*20.02%*	*20.02%*	*20.02%*	*20.02%*	
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%*	

Table 2: 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). "+ OOV" indicates the pruning technique combined with out-of-vocabulary clustering for token remapping. Parameter reduction percentages show total model size decrease for each method.

Implementation Details All experiments were conducted on an NVIDIA RTX 4090 GPU. The vocabulary pruning techniques described in this paper were implemented in a public repository[1]. This implementation includes all pruning methods, evaluation scripts, and analysis tools described in this work. For encoder layer pruning, a custom fork of the LoSparse implementation was used[2].

Results Table 2 presents a comprehensive evaluation across GLUE tasks for different the 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 (+2.4 percentage points on average).

Pruning effectiveness varies by task: attention-based methods excel on single-sentence tasks and paraphrase detection, while TF-IDF-based methods perform better on complex reasoning tasks.

As shown in Figure 2, vocabulary pruning outperforms encoder-layer pruning (LoSparse) at lower pruning ratios (5-20%), while requiring only minimal computational overhead as an offline pre-fine-tuning method. A 20% reduction in total parameters (77.34% of vocabulary size) maintains competitive performance, making this approach particularly valuable for resource-constrained environments. 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.

[1] `https://github.com/WoutDeRijck/vocab-pruning.git`
[2] `https://github.com/WoutDeRijck/LoSparse.git`

The benchmark results in Table 3 show 20% storage reduction and 14.83% lower GPU memory use with negligible inference time impact (+1.33%).

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.

Conclusion

In this work, we designed and evaluated a series of vocabulary pruning techniques for the ModernBERT language model to reduce model size while maintaining task performance.

We found that simple vocabulary pruning techniques outperform encoder-layer pruning for moderate compression ratios. Our TF-IDF-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 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 assumption that encoder pruning should precede 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.

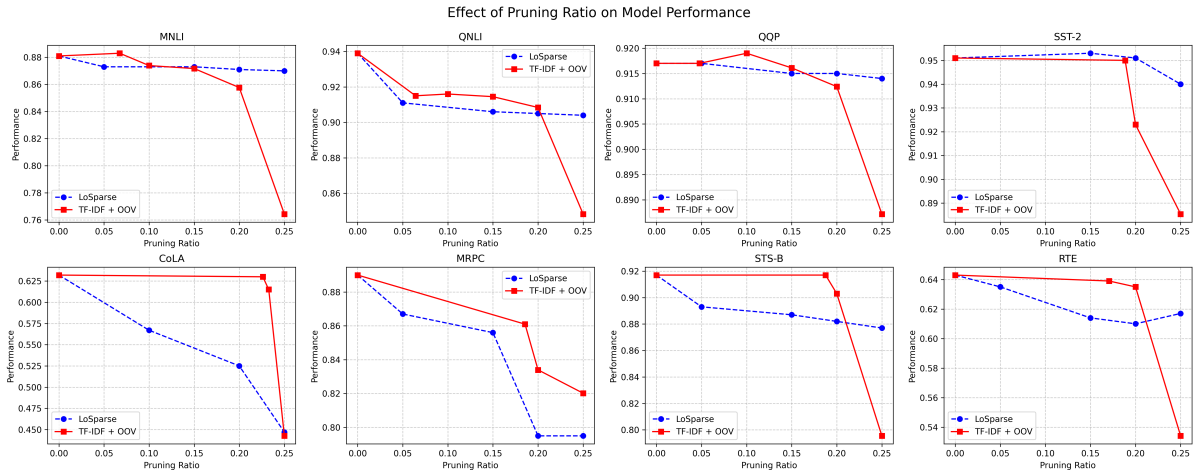

Figure 2: Performance comparison between vocabulary pruning (TF-IDF + OOV) and encoder pruning (LoSparse) across different pruning ratios for MNLI, QNLI, and QQP tasks.

Future Work

Future research could explore an integrated approach combining vocabulary pruning with encoder-layer compression techniques like LoSparse, as preliminary experiments suggest these approaches are complementary. We hypothesize that pruning up to a certain threshold using this combined approach will yield better results than using either vocabulary pruning or encoder-layer pruning alone.

Additionally, evaluating vocabulary pruning on information retrieval tasks may reveal different optimal pruning strategies, as retrieval models likely depend on different token importance distributions than classification tasks.

References

[1] Amine Abdaoui, Arnaud Fehrentz, and Thomas Lavergne. Load what you need: BERT optimized for speed. *arXiv preprint arXiv:2007.02450*, 2020.

[2] Timothée Bousquet et al. PEMA: Efficient fine-tuning by partial embedding matrix adaptation. *arXiv preprint arXiv:2303.00868*, 2023.

[3] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *ICLR*, 2020.

[4] Zixuan Li, Meiqi Ding, Zhou Yu, Tuo Zhao, and Tengyu Ma. Losparse: Structured pruning via low-rank and sparse decomposition. In *International Conference on Learning Representations (ICLR)*, 2023.

[5] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *NeurIPS*, 2019.

[6] 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.

[7] Raghuveer 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.

[8] 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.

[9] Yelong Shen et al. TextPruner: A unified framework for pruning token-level and feature-level redundancy in text. In *EMNLP*, 2022.

[10] 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.

[11] 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.

[12] Yifei Wang et al. LightToken: Structured embedding compression for transformer inference. In *NeurIPS*, 2023.