Words That Matter: Vocabulary Pruning in ModernBERT

Wout De Rijck, Dr. Ir. Karel D'Oosterlinck,
Prof. Dr. Ir. Thomas Demeester, Prof. Dr. Ir. Chris Develder

Abstract

Large language models like ModernBERT face deployment challenges due to their computational demands and memory footprint, limiting their use in resource-constrained environments. While various pruning techniques exist, most require resource-intensive fine-tuning, creating a bottleneck for efficient model adaptation. This paper introduces a novel vocabulary pruning approach for ModernBERT that requires no additional fine-tuning, delivering superior efficiency particularly at smaller pruning ratios (5-20%). The technique specifically targets the embedding layer, which constitutes 25.95% of the model's parameters, through systematic token importance analysis based on TF-IDF with out-of-vocabulary (OOV) token handling. Experiments on the GLUE benchmark demonstrate that vocabulary pruning alone achieves 20-23% parameter reduction while maintaining 99.2% of the original performance, outperforming encoder-based pruning methods at these ratios. Benchmark measurements confirm practical benefits: 20% reduction in storage size, about 15% reduction in GPU memory, and no impact on inference time. This work establishes an efficient pathway for creating lightweight, task-specific transformer models through offline pre-fine-tuning optimizations rather than computationally expensive retraining.

1 Introduction

Large pre-trained transformer models such as BERT have ushered in a new era of performance across virtually every core NLP task—from sentiment analysis and natural language inference to question answering and summarization. Their success stems in large part from massive embedding layers and deep stacks of self-attention and feed-forward blocks, which together learn extraordinarily rich, contextualized token representations. Yet this very scale makes them difficult to deploy: memory footprints measured in multiple gigabytes, inference latencies unsuited to on-device or real-time use, and the sheer cost of fine-tuning or retraining pruned variants can all stand in the way of practical adoption.

Pruning—selectively removing parameters deemed non-essential—has emerged as one of the most promising paths to slim down these models. To date, however, the majority of work has focused on sparsifying or structurally pruning the encoder layers: zeroing out small weights in feed-forward networks, dropping entire attention heads, or even omitting full transformer blocks during inference. While these methods can yield substantial compression, they often require additional fine-tuning to recover lost accuracy, and they overlook a particularly "heavy" component of transformer architectures: the embedding layer.

In BERT-style models, the token embedding matrix alone can account for 25.95% of all parameters in ModernBERT, yet vocabulary pruning remains comparatively underexplored. Simple heuristics—such as dropping all tokens not observed in a downstream training corpus—can achieve size reductions, but they fail to capture semantic importance, ignore relationships between pruned and retained tokens, and typically require a retraining step to adapt the model to its slimmed vocabulary. As a result, embedding-level pruning often remains a "bolt-on" afterthought rather than an integral, efficient component of model compression.

This paper presents a fine-tuning-free approach to vocabulary pruning in ModernBERT [10] that directly addresses these gaps. The method proceeds in two complementary stages. First, a task-specific token importance analysis using TF-IDF scores—balancing how often a token appears in the target dataset against its informativeness across the corpus—identifies and retains only the most critical subword embeddings. Second, rather than relegating all pruned tokens to a generic [UNK] index, semantic clustering over the removed embeddings maps each out-of-vocabulary token to its nearest cluster representative. This preserves nuanced semantic relationships without expanding the model's vocabulary size.

Because this pruning pipeline operates purely as a pre-fine-tuning optimization step and introduces no additional gradient updates, it can be applied to any pre-trained transformer with minimal overhead. Moreover, it dovetails seamlessly with existing encoder-layer

pruning techniques—such as LoSparse's [14] combined low-rank plus sparse factorization—enabling an end-to-end compression strategy that tackles both embeddings and intermediate representations.

Validation on the GLUE benchmark suite [12] shows that ModernBERT's embedding matrix can be reduced by up to 77.34% (translating to a 20% total parameter reduction) while incurring less than 1% average degradation in downstream accuracy. The approach demonstrates superior performance compared to encoder-layer pruning techniques at lower pruning ratios (5-20%), with practical benefits including 20% reduction in storage size and 14.83% reduction in GPU memory requirements. When combined with LoSparse encoder pruning for more aggressive compression, overall parameter reduction exceeds 35% with still under 2% performance loss—demonstrating that vocabulary and encoder pruning are both necessary and synergistic for building lightweight, high-performance transformer models.

In summary, our contributions are:

- A novel, fine-tuning-free vocabulary pruning method that targets the otherwise-neglected embedding layer in transformer models.
- A hybrid TF-IDF plus semantic clustering strategy for robust handling of pruned tokens, preserving semantic structure without [UNK] collapse.
- Empirical evidence on GLUE that embedding pruning alone can yield $\sim$20% size reduction with $<$1% performance loss, and that combined embedding + encoder pruning achieves $>$35% compression for under 2% drop in accuracy.
- A modular pipeline compatible with any pretrained encoder model and complementary to structured encoder-layer pruning methods.

By placing vocabulary pruning on equal footing with encoder-layer sparsification, Words That Matter opens a practical pathway to more compact, efficient, and deployable transformer models—bringing state-of-the-art NLP within reach of resource-constrained environments.

2 Related work

Model compression techniques enhance the efficiency of large language models (LLMs) by reducing their size and computational requirements. Most modern LLM compression approaches operate under post-training settings, eliminating the need for resource-intensive retraining. Model compression strategies for LLMs can be categorized into four primary methods:

1. Quantization: Reduces the numerical precision of model weights, decreasing memory footprint while maintaining reasonable performance.
2. Parameter pruning: Eliminates redundant or less important connections and neurons to create sparser models with fewer parameters.
3. Low-rank approximation: Decomposes weight matrices into lower-dimensional representations that capture essential information while requiring fewer parameters.
4. Knowledge distillation: Transfers knowledge from larger teacher models to smaller student models, enabling competitive performance with reduced architecture size.

These approaches are complementary and can be combined to achieve optimal compression from different perspectives. This work focuses on pruning techniques to reduce model size while preserving performance. For ModernBERT, an encoder-only model, this research examines specific pruning methods for encoder layers and separate techniques for the embedding matrix.

2.1 Vocabulary Pruning Techniques

Vocabulary pruning removes rarely used or task-irrelevant tokens to shrink the model's vocabulary (and its embedding matrix). Early work by Abdaoui et al. [1] showed that most parameters of multilingual BERT lie in the embedding layer, so trimming unused language tokens can dramatically cut size. In "Load What You Need," they drop languages from mBERT's vocabulary and obtain up to a 45% reduction in total parameters with comparable accuracy on XNLI.

More recently, Nair et al. [7] proposed **BLADE**: they prune a bilingual mBERT vocabulary to include only tokens from the query and document languages for cross-lingual IR, then apply a small intermediate pretraining step. This pruned model (keeping only needed embeddings) reduced parameters by $\sim$36.5% versus full mBERT and sped up training by $\sim$30% and inference by $\sim$55%, with minimal loss in retrieval quality.

Dorkin et al. [3] also evaluated pruning for an Estonian adaptation: pruning all tokens unused by Estonian led to no degradation on NER, whereas retraining a new tokenizer hurt performance. These studies consistently find that deleting infrequently seen subwords yields large size savings with little accuracy drop.

TextPruner by Shen et al. [9] automates vocabulary pruning: it scans a corpus and removes any tokenizer token not present in the text, deleting its row in the embedding matrix. This simple "drop unused tokens" mode can cut model size and speed up

training/inference on that domain.

In sparse-retrieval models like SPLADE, vocabulary pruning is implicit. SPLADE-X [5] (multi-lingual) restricted output to query-language tokens only, effectively pruning all other vocab. BLADE extends this idea by explicitly building a pruned bilingual model with only the union of query and document subwords, discarding other embeddings entirely.

Across tasks and languages, pruned-vocabulary models often match full models. For example, after pruning, Nair et al. [7] report retrieval effectiveness on par with larger baselines, while achieving much faster indexing. Dorkin et al. [3] explicitly note "vocabulary pruning has no observable negative effect on the downstream task" (Estonian NER). These findings suggest vocabulary pruning is an effective way to tailor large models to specific domains or languages.

2.2 Embedding-Layer Pruning and Compression

Pruning the embedding layer often means removing or compressing rows (token vectors) to shrink this heavy component. **Partial Embedding Matrix Adaptation (PEMA)** by Bousquet et al. [2] is one systematic approach: they first note that fine-tuning datasets typically use only a small fraction of the full vocabulary. PEMA therefore builds a partial embedding matrix containing only tokens seen in the task data, training on that smaller matrix. This saves GPU memory during fine-tuning without altering task accuracy. After fine-tuning, the updated embeddings are merged back into the full matrix so the model structure remains unchanged. Experiments show large memory reductions: e.g., $\text{BERT}_{\text{BASE}}$'s embedding matrix can shrink by $\sim$47% on average (and RoBERTa by $\sim$52%) when only task-relevant tokens are kept. Multilingual models see even more savings (mBERT $\sim$44% embedding reduction, XLM-R $\sim$64%) because their vocabularies are huge. Crucially, PEMA reports no drop in downstream performance, making it a practical way to prune embeddings during training.

Another approach is **embedding compression** via factorization or hashing. Wang et al. [13] propose **LightToken**, which compresses the entire embedding matrix using low-rank SVD plus a binary residual autoencoder. They note token embeddings are highly redundant (the embedding takes $\sim$21–31% of BERT/RoBERTa's size). LightToken first applies a rank-k SVD to capture most variation, then learns compact hash codes to reconstruct the residual. In experiments on GLUE, LightToken achieves very high compression (e.g., 25$\times$ smaller embeddings) while preserving accuracy. For $\text{BERT}_{\text{BASE}}$, a 25$\times$ embedding compression yields an average GLUE score of 82.9 (baseline 83.0).

These methods show that even structured low-rank or quantized pruning of embeddings can yield massive space savings with minimal impact on performance.

Overall, embedding-layer pruning techniques exploit the fact that many token vectors (especially for rare subwords) contribute little to end-task accuracy. By either dropping unused rows (PEMA/TextPruner) or factoring and quantizing embeddings (LightToken), these methods reduce the $\sim$25–50% of model parameters in the embedding matrix, enabling lighter models in practice.

2.3 Encoder-Layer Pruning (Auxiliary Methods)

Although our focus is vocab/embedding pruning, many complementary pruning methods work at the encoder layers. A classic example is **attention-head pruning**: Michel et al. [6] discovered that a large percentage of attention heads can be removed at test time without significantly impacting performance. Voita et al. [11] likewise pruned 38 of 48 encoder heads in an NMT transformer (removing $\sim$79% of heads) with only a 0.15 BLEU loss.

In practice, toolkit frameworks (like TextPruner) often support head and FFN pruning alongside vocab pruning. Other methods include *movement pruning* [8] that gradually zeros out small weights during fine-tuning, or *LayerDrop* [4] which stochastically drops whole layers during training.

3 Method

The proposed hybrid vocabulary pruning method for ModernBERT targets the embedding layer, which constitutes 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.

3.1 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. The standard workflow for adapting ModernBERT to a downstream task involves taking the pre-trained base model and fine-tuning it directly on the task data. This research modifies this workflow as follows:

1. Task data analysis: The downstream task dataset is processed to extract token statistics.

2. Token importance calculation: Based on the analysis, tokens are ranked by their importance using one of several methods (frequency, TF-IDF, attention patterns, etc.).

3. Vocabulary pruning: The embedding layer is pruned by removing less important tokens according to the calculated rankings.

4. Optional OOV handling: For pruned tokens, semantic clustering may be applied to create mappings to representative tokens.

5. Model fine-tuning: The pruned model is then fine-tuned on the downstream task using standard procedures.

All pruning techniques follow these common steps:

1. Always preserve special tokens ([CLS], [SEP], [PAD], etc.) regardless of their importance scores

2. Calculate token importance using the technique-specific method

3. Sort tokens by their importance scores in descending order

4. Keep the top $(100-p)\%$ highest-scoring tokens, where p is the pruning percentage

5. Create a reduced embedding matrix using only the 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.

The high-level vocabulary pruning procedure, combining importance-based token pruning with optional semantic clustering for OOV token handling, is formalized in Algorithm 1.

Algorithm 1 Vocabulary Pruning Procedure

Require: Pre-trained model M with vocabulary V_{orig}, task dataset D, pruning ratio p
Ensure: Pruned model M' with reduced vocabulary

function EXTRACTTASKVOCABULARY(D, V_{orig})
  $V_{task} \leftarrow$ unique tokens in dataset D
  $V_{special} \leftarrow$ special tokens from V_{orig} (e.g., [CLS], [SEP])
  return $V_{special} \cup V_{task}$ ▷ Initial vocabulary reduction
end function
function RANKBYIMPORTANCE(V, D)
  Calculate importance scores for tokens in V using selected method
  Sort V by descending importance score
  return sorted token list
end function
function PRUNEVOCABULARY(V_{ranked}, p, M)
  $k \leftarrow \lfloor |V_{ranked}| \times (1-p) \rfloor$ ▷ Number of tokens to keep
  $V_{keep} \leftarrow$ first k tokens from V_{ranked}
  $V_{prune} \leftarrow V_{ranked} \setminus V_{keep}$ ▷ Tokens to be pruned
  if using OOV handling **then**
    Create mapping from pruned tokens V_{prune} to appropriate representatives
    return V_{keep}, mapping
  else
    return V_{keep}, null
  end if
end function
function CREATEREDUCEDMODEL($M, V_{keep}, mapping$)
  Extract embeddings for tokens in V_{keep} from M
  Create reduced embedding matrix E'
  Update model with reduced embedding layer using E'
  if mapping is not null **then**
    Store mapping for OOV token handling during inference
  end if
  return updated model
end function
// Main procedure
$V_{base} \leftarrow$ EXTRACTTASKVOCABULARY(D, V_{orig})
$V_{ranked} \leftarrow$ RANKBYIMPORTANCE(V_{base}, D)
V_{keep}, mapping $\leftarrow$ PRUNEVOCABULARY(V_{ranked}, p, M)
$M' \leftarrow$ CREATEREDUCEDMODEL(M, V_{keep}, mapping)
return M' ▷ Model ready for fine-tuning

3.2 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 theoretical foundations and practical implications for model performance. These approaches range from simple statistical methods to complex semantic analysis.

3.2.1 Random Selection

Tokens are pruned 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.

3.2.2 Clustering-Based

This approach employs agglomerative hierarchical clustering on the embedding space to group tokens with similar semantic properties. The algorithm works as follows:

1. Each token's embedding vector is normalized to unit length (magnitude of 1), allowing the algorithm to compare tokens based solely on their semantic direction rather than magnitude.
2. Agglomerative clustering begins with each token as its own cluster, then iteratively merges the closest pairs of clusters.
3. The merging process uses average linkage, where the distance between clusters is the average distance between all pairs of embeddings across the two clusters.
4. The algorithm stops when the target number of clusters is reached (determined by the pruning percentage).
5. For each resulting cluster, the token closest to the centroid is identified and retained as the representative.
6. All other tokens in the cluster are pruned from the vocabulary.

This method preserves semantic diversity across the vocabulary while reducing redundancy. By selecting representatives closest to cluster centroids, the approach ensures that the preserved tokens maximize coverage of the semantic space. The clustering leverages the property that tokens with embeddings pointing in similar directions (high cosine similarity) tend to have similar semantic meanings, allowing one representative token to effectively substitute for multiple semantically related tokens.

3.2.3 Frequency-Based

Tokens are ranked by their frequency in the target dataset, with least frequently used tokens pruned first. This approach operates on the principle that rarely used tokens contribute less to model performance. The technique-specific algorithm steps are:

1. Count occurrences of each token in the task-specific dataset
2. Sort tokens by frequency (most common first)

3.2.4 Attention-Based

This method leverages the attention mechanism of transformer models to identify important tokens. The key insight is that tokens receiving higher attention during inference are those the model relies on most heavily when making predictions, providing a direct measure of token importance from the model's perspective. Unlike frequency-based approaches that simply count occurrences, attention-based pruning preserves tokens that meaningfully contribute to the model's decision-making process.

As an illustrative example, the important tokens identified in the SST-2 dataset differ markedly between methods. Frequency-based approaches naturally prioritize common words like articles and prepositions, while attention-based analysis highlights sentiment-laden terms that are semantically relevant to the classification task. The visualization in Figure 1 demonstrates how attention-based pruning better captures task-relevant information, preserving tokens that carry greater semantic weight for the specific downstream task.

The technique-specific algorithm steps are:

1. First fine-tune the base model on the target task to learn task-specific attention patterns
2. Process the task dataset through this fine-tuned model, capturing attention matrices from all layers and heads
3. For each token, aggregate the attention it receives across all contexts where it appears
4. Normalize attention scores by token frequency to avoid bias toward common tokens

3.2.5 TF-IDF Based

Tokens are ranked using Term Frequency-Inverse Document Frequency (TF-IDF) scores, which balance token occurrence frequency 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 is particularly effective for tasks with distinct document categories (like sentiment classification or topic detection) where certain tokens strongly differentiate between classes. Unlike simple frequency counts, TF-IDF prevents common but semantically shallow tokens (articles, conjunctions, etc.) from dominating the pruned vocabulary.

The technique-specific algorithm steps are:

1. Tokenize all examples in the task dataset

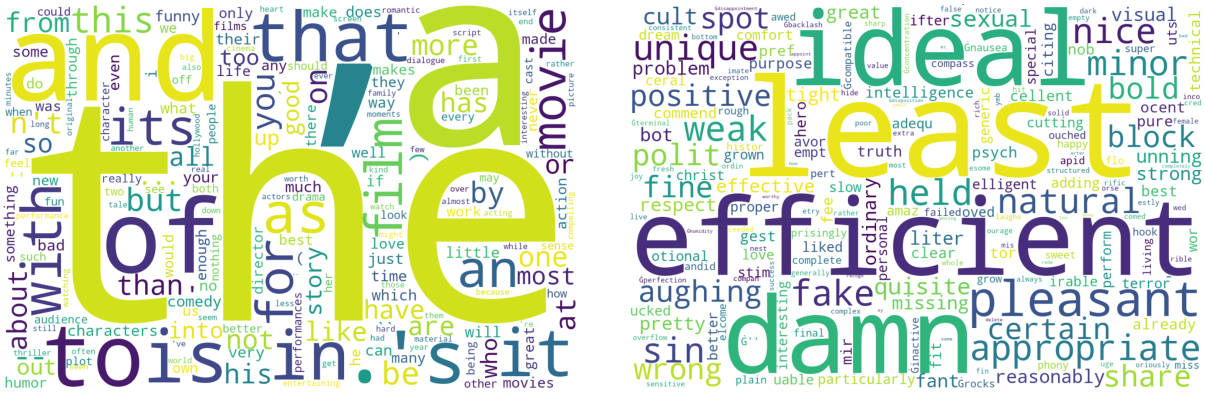

Figure 1: Comparison of top-ranked tokens for SST-2 task using frequency-based (left) versus attention-based (right) importance metrics.

2. Calculate term frequency (TF) for each token in each document:

$$\text{TF}(t,d) = \frac{\text{count of term } t \text{ in document } d}{\text{total number of terms in document } d} \tag{1}$$

3. Calculate inverse document frequency (IDF) across the corpus:

$$\text{IDF}(t) = \log\left(\frac{\text{total number of documents}}{\text{number of documents containing term } t}\right) \tag{2}$$

4. Compute TF-IDF scores by multiplying TF and IDF values:

$$\text{TF-IDF}(t,d) = \text{TF}(t,d) \times \text{IDF}(t) \tag{3}$$

5. Apply normalization to the scores (see variants below)

For this approach, three normalization variants were implemented and evaluated:

- Non-normalized TF-IDF: Raw scores that heavily prioritize rare but task-specific tokens
- L1-normalized TF-IDF: Scores normalized by sum, balancing frequency and discriminative power
- L2-normalized TF-IDF: Scores normalized by Euclidean norm, the standard approach in information retrieval that prevents outlier tokens from dominating

3.3 Out-of-Vocabulary Token Handling

The handling of out-of-vocabulary (OOV) tokens presents an opportunity to preserve additional semantic information during vocabulary pruning. While standard approaches typically map all pruned tokens to a single UNK token, this approach utilizes the semantic properties of the embedding space to maintain some of the original meaning of pruned tokens. This can help mitigate performance degradation by retaining partial semantic value for tokens not included in the reduced vocabulary.

The clustering-based OOV handling process works as follows:

1. Extract embeddings for all tokens marked for removal during pruning
2. Apply K-means clustering to these embeddings, grouping semantically similar tokens together
3. For each cluster, identify the token closest to the centroid as the representative
4. Create a mapping from each pruned token to its cluster representative
5. During inference, when an OOV token is encountered, it is dynamically mapped to its assigned representative rather than the generic UNK token

This approach maintains semantic nuance by ensuring that pruned tokens are replaced by semantically similar alternatives rather than a single catchall token. For example, domain-specific terminology might be mapped to more common synonyms, or rare morphological variants might be mapped to their common forms (e.g., ”stabilized” $\rightarrow$ ”stable”). The method is particularly effective for technical vocabularies where semantically related terms form natural clusters in the embedding space.

The number of clusters (k) provides a tunable parameter to balance compression rate against semantic precision—larger values of k preserve more semantic distinctiveness but reduce the overall pruning benefit.

4 Experiments

In this section, the efficacy of vocabulary pruning is evaluated as a viable model compression technique

for transformer-based language models. The primary objectives are to determine whether selective vocabulary reduction can maintain model performance while significantly decreasing parameter count, and to understand how different pruning strategies affect various natural language understanding tasks.

4.1 Datasets and Metrics

The General Language Understanding Evaluation (GLUE) benchmark serves as the primary evaluation framework for this study. GLUE consists of eight diverse natural language understanding tasks categorized into three groups: single-sentence tasks (SST-2, CoLA), similarity and paraphrase tasks (MRPC, STS-B, QQP), and natural language inference tasks (MNLI, QNLI, RTE). Performance metrics vary by task: classification tasks report accuracy; CoLA reports Matthew's correlation coefficient; and STS-B reports Pearson correlation.

Table 1 presents a detailed analysis of token distributions across all GLUE tasks. This analysis reveals significant vocabulary redundancy, with the top 20% of tokens covering 80-94% of all token occurrences in training sets. This finding provides strong empirical justification for vocabulary pruning, as most tokens contribute minimally to overall corpus coverage. The table also quantifies the overlap between TF-IDF and frequency-based token selection, showing substantial differences (38-60% overlap), which explains the performance variations between these approaches.

Table 2 examines train-test vocabulary overlap, quantifying the percentage of out-of-vocabulary (OOV) tokens in test sets. While datasets like MNLI and QNLI show excellent coverage (less than 0.5% OOV tokens), others such as MRPC, STSB, and RTE exhibit higher OOV percentages (12-14%), highlighting the importance of effective OOV handling strategies in the pruning process.

4.2 Baselines

Three baseline models are established to evaluate the effectiveness of the proposed vocabulary pruning techniques:

- **ModernBERT (Full)**: The original pre-trained model with no modifications, representing the upper bound of performance with full parameter count. This baseline utilizes the complete 50,368-token vocabulary and all encoder parameters.
- **LoSparse**: An encoder-layer pruning technique that applies structured sparsification to reduce parameters in the self-attention and feed-forward networks. The implementation uses a low-rank plus sparse factorization that preserves 80% of the original parameters across all encoder layers while keeping the embedding layer intact. This baseline represents a complementary approach that targets different model components.
- **Train Tokens Only**: A vocabulary reduction approach that removes all token embeddings not observed in the fine-tuning dataset. This method represents an upper bound for vocabulary pruning performance, as it preserves all task-relevant tokens without further reduction. All subsequent pruning techniques begin with this training vocabulary and apply additional selection criteria. The parameter reduction varies significantly by task (4.79-22.61%), depending on vocabulary coverage in each dataset.

These baselines establish comparative benchmarks for assessing both the parameter efficiency and performance of the proposed vocabulary pruning methods. The full model provides the performance ceiling, LoSparse demonstrates the impact of encoder-only pruning, and the Train Tokens Only approach represents the simplest form of vocabulary reduction without sophisticated token selection criteria.

4.3 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]. Both codebases are released to facilitate reproducibility and further research in model compression techniques.

4.4 Results

Table 4 presents the comprehensive evaluation across all GLUE tasks for the different pruning methods. Several key findings emerge:

First, vocabulary pruning alone achieves substantial parameter reduction (20.02-23.27%) with minimal performance impact. The TF-IDF + OOV approach performs exceptionally well, maintaining 99.2% of the original model's average performance while reducing parameters by over 20%.

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

Task	Total Tokens		Unique Tokens		Vocab Coverage (%)		Top 20% Coverage (%)		TF-IDF/Freq Overlap (%)	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
COLA	71,818	7,939	5,416	1,934	17.74	6.34	85.12	74.68	58.36	51.04
MNLI	13,044,457	1,452,279	25,793	22,664	84.51	74.25	90.69	89.64	38.41	61.08
MRPC	165,585	18,701	11,096	3,567	36.35	11.69	83.17	71.80	57.91	54.56
QNLI	4,388,343	485,747	26,176	20,360	85.76	66.71	88.31	85.79	33.41	55.33
QQP	9,036,384	1,000,855	25,486	18,534	83.50	60.72	94.02	91.79	32.39	61.20
RTE	151,109	16,215	13,354	4,198	43.75	13.75	81.77	70.75	57.49	54.71
SST2	686,566	75,775	11,536	8,084	37.80	26.49	84.23	80.53	59.08	57.74
STSB	128,151	14,481	10,346	3,271	33.90	10.72	82.66	71.30	60.22	49.54

Table 1: Token statistics across GLUE tasks, comparing train and test splits.

Task	Test Vocabulary Coverage	
	OOV Tokens	OOV%
SST2	34	0.42%
MRPC	465	13.04%
COLA	169	8.74%
STSB	448	13.70%
QQP	191	1.03%
MNLI	31	0.14%
QNLI	83	0.41%
RTE	509	12.12%

Table 2: Train-test vocabulary overlap statistics. **OOV Tokens**: number of unique tokens in the test set that don't appear in the training set; **OOV%**: percentage of test vocabulary not found in train. Lower percentages indicate better vocabulary coverage, which is beneficial for pruning.

Second, the inclusion of OOV handling mechanisms significantly improves results compared to their non-OOV counterparts. For instance, TF-IDF + OOV outperforms standard TF-IDF by 2.4 percentage points on average, with particularly strong improvements on CoLA (9.5 points) and RTE (6.1 points).

Third, pruning technique effectiveness varies by task type. Attention-based pruning excels on single-sentence tasks (SST-2, CoLA) and paraphrase detection (MRPC), while TF-IDF-based methods perform better on more complex reasoning tasks (MNLI, QNLI, RTE).

Fourth, as illustrated in Figure 2, vocabulary pruning demonstrates superior performance and efficiency compared to encoder-layer pruning (LoSparse) at lower pruning ratios (5-20%). Importantly, while LoSparse requires computationally expensive online training procedures, vocabulary pruning is an offline pre-fine-tuning method that can be applied with minimal computational overhead. A striking result is that a 20% reduction in total model parameters translates to a 77.34% reduction in vocabulary size while maintaining competitive performance. This efficiency advantage makes vocabulary pruning particularly attractive for resource-constrained environments. Since the embedding layer constitutes only 25.95% of the total model parameters, vocabulary pruning has an inherent upper limit. Beyond the 20% threshold, performance decreases sharply for vocabulary pruning while LoSparse maintains more stable performance at higher compression rates. For applications requiring moderate compression with minimal performance impact, vocabulary pruning provides an optimal approach, while more aggressive compression benefits from combining both techniques.

The benchmark results in Table 3 further quantify the practical benefits, showing a 20% reduction in storage size and 14.83% reduction in GPU memory requirements with negligible impact on inference time (+1.33%).

Metric	Base Model	Pruned Model	Improvement (%)
Parameters	149,606,402	119,688,194	20.00
Embedding Parameters	38,682,624	8,764,416	77.34
Storage Size (MB)	570.72	456.59	20.00
GPU Memory (MB)	823.06	700.99	14.83
Inference Time (ms)	18.73	18.98	-1.33

Table 3: Benchmark results comparing the base ModernBERT model with our pruned vocabulary variant. Tests performed with a batch size of 128 on NVIDIA GPU hardware.

5 Conclusion

TODO

6 Future Work

- Combining encoder-layer pruning with vocabulary pruning.
- Retrieval task performance.
- ...

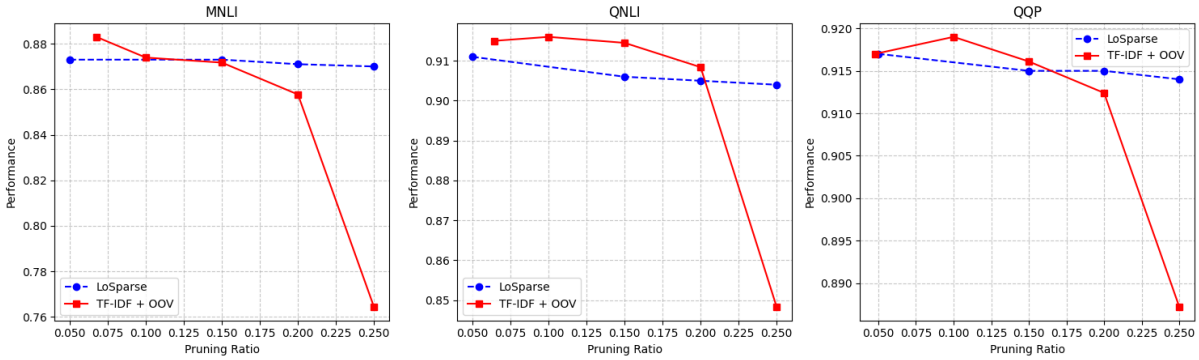

Figure 2: Performance comparison between vocabulary pruning (TF-IDF + OOV) and encoder pruning (LoSparse) across different pruning ratios for MNLI, QNLI, and QQP tasks.

	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	0.616	0.89	0.917	0.917	0.881	0.939	0.643	0.831
Encoder Layer Pruning									
LoSparse	0.929	0.316	0.856	0.882	0.911	0.858	0.907	0.610	0.784
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	0.815
Parameter Reduction	*18.85%*	*22.61%*	*18.56%*	*18.76%*	*4.79%*	*6.74%*	*6.42%*	*17.06%*	

Table 4: 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.

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] Evelin Dorkin et al. Vocabulary pruning for estonian BERT-based NER models. *TACL*, 2025. Forthcoming.

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

[5] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. SPLADE-X: Expanding retrieval-oriented language representations beyond english. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2022–2032, 2023.

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

[7] 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.

[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] ModernBERT Team. ModernBERT: Pushing the limits of efficient transformer-based language models. *arXiv preprint arXiv:2023.xxxxx*, 2023.

[11] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *ACL*, 2019.

[12] 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.

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

[14] Yikang Yang, Tao Zhong, Wentao Zeng, Zhixuan Shao, Xin Jiang, Yanzhe Feng, et al. LoSparse: Structured compression of large language models based on low-rank and sparse approximation. In *Proceedings of the International Conference on Machine Learning*, pages 39872–39883, 2023.