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. We introduce a novel vocabulary pruning approach for ModernBERT that requires no additional fine-tuning, delivering superior efficiency particularly at smaller pruning ratios. Our technique specifically targets the embedding layer, which constitutes approximately 25% of the model's parameters, through a systematic token importance analysis. We demonstrate that combining our vocabulary pruning method with LoSparse techniques for encoder layers achieves comprehensive model compression with minimal performance degradation. Experimental results show our approach reduces model size by up to X% while maintaining Y% of the original performance across standard benchmarks. This work establishes a practical pathway for creating lightweight, task-specific transformer models without the computational burden of traditional pruning methods.

1 Introduction

1. Introduction of the ModernBERT paper.
2. Architecture of ModernBERT.

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 Encoder Layer Pruning

2.1.1 LoSparse

Low-Rank and Sparse approximation (LoSparse) introduces a model compression strategy that decomposes each weight matrix W as the sum of a low-rank component (UV) and a sparse component (S). The low-rank part, computed via singular value decomposition, extracts the shared, coherent features among neurons, while the sparse part targets and removes redundant, incoherent features. This dual approach overcomes the limitations of traditional methods by avoiding the excessive pruning of expressive neurons and preserving diversity in the representation. Extensive experiments on natural language understanding, question answering, and generation tasks show that LoSparse achieves significant parameter reduction all while maintaining competitive performance levels.

2.1.2 Other encoder layer pruning techniques

TODO

2.2 Vocabulary Pruning

While encoder layer pruning techniques like LoSparse have shown promising results, the embedding layer presents a unique challenge in transformer models. Accounting for approximately 25% of the total parameters, the vocabulary embedding layer offers a substantial opportunity for model compression. However, unlike encoder layer pruning, vocabulary reduction requires careful consideration of semantic relationships and token importance to maintain model performance.

2.2.1 Direction is what you need

2.2.2 TF-IDF

2.2.3 PEMA

Partial Embedding Matrix Approximation (PEMA) makes finetuning efficient by removing the tokens in the embedding layers that are not used in the current task.

2.2.4 LightToken

LightToken is a lightweight token embedding framework that produces compressed token embeddings in a task and model-agnostic fashion. It's designed to be compatible with different architectures and applicable to any downstream task. The framework integrates low-rank approximation, a novel residual binary autoencoder, and a new compression loss function to significantly improve model compression ratio without sacrificing performance.

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 removed tokens can be mapped to semantically similar ones instead of a single unknown token. The method consists of three main components: token importance analysis, selective pruning, and semantic clustering for out-of-vocabulary (OOV) tokens.

3.1 Token Importance Analysis

Multiple approaches for token importance estimation were systematically evaluated:

1. **Random Selection**: Tokens are pruned randomly without consideration for importance, serving as a baseline approach.

2. **Frequency-Based**: Tokens are ranked by their frequency in the target dataset, with least frequently used tokens pruned first. This approach assumes that rarely used tokens contribute less to model performance.

3. **Clustering-Based**: This approach employs agglomerative or k-means clustering on the embedding space to group tokens with similar semantic properties. For each cluster, the token closest to the centroid is retained as a representative, while others are pruned. This preserves semantic diversity across the vocabulary while reducing redundancy.

4. **Attention-Based**: This method uses attention patterns from fine-tuning to determine token importance. For each token, attention scores across all layers and heads are aggregated to capture its contextual relevance within the specific task. Tokens receiving consistently higher attention are considered more important for model decisions, providing a more nuanced importance measure that captures both frequency and semantic significance.

5. **TF-IDF Based**: Tokens are ranked using Term Frequency-Inverse Document Frequency scores, which balance token occurrence frequency with discriminative power across documents.

The TF-IDF approach demonstrated superior performance and was adopted as the primary method. For this approach, three normalization variants were tested:

- Non-normalized TF-IDF (prioritizing rare but task-specific tokens)

- L1-normalized TF-IDF (balancing frequency and importance)

- L2-normalized TF-IDF (standard approach, providing optimal results)

The TF-IDF method effectively identifies task-relevant tokens by assigning higher importance to tokens that frequently appear in specific documents but are less common across the entire corpus. This enables selective vocabulary pruning while preserving tokens critical for maintaining task performance.

3.2 Hybrid Pruning Method

The hybrid approach combines importance-based token pruning with semantic clustering for OOV token handling. The method is formalized in Algorithm 1.

Algorithm 1 Hybrid Vocabulary Pruning

Require: Model M with vocab V, dataset D, pruning % p, clusters k, method *method*
Ensure: Pruned model M' with reduced vocabulary

// Stage 1: Calculate token importance using selected method (random, frequency, attention, or TF-IDF)
$importance \leftarrow \text{CalculateTokenImportance}(V, D, M, method)$
// Stage 2: Prune lowest-importance tokens based on pruning percentage
$tokens\_sorted \leftarrow \text{SortByImportance}(V, importance)$
$keep\_count \leftarrow |V| \times (1 - p/100)$
$tokens\_to\_keep \leftarrow tokens\_sorted[1 : keep\_count]$
$tokens\_to\_remove \leftarrow V \setminus tokens\_to\_keep$
// Stage 3: Cluster removed tokens and identify representatives for OOV mapping
$embeddings \leftarrow M.embedding\_layer.weight$
$removed\_embeddings \leftarrow embeddings[tokens\_to\_remove]$
$kmeans \leftarrow \text{KMeans}(n\_clusters = k)$
$clusters \leftarrow kmeans.fit\_predict(removed\_embeddings)$
$oov\_map \leftarrow \{\}$; $centers \leftarrow \{\}$
for each cluster c in $range(k)$ **do**
  $indices \leftarrow \{i : clusters[i] = c\}$
  if *indices* is not empty **then**
    $centroid \leftarrow kmeans.cluster\_centers_{[c]}$
    $distances \leftarrow \{\|removed\_embeddings[i] - centroid\| : i \in indices\}$
    $closest\_idx \leftarrow \arg\min(distances)$
    $center\_token \leftarrow tokens\_to\_remove[indices[closest\_idx]]$
    $centers.add(center\_token)$
    for $idx \in indices$ **do**
      $oov\_map[tokens\_to\_remove[idx]] \leftarrow center\_token$
    end for
  end if
end for
// Stage 4: Create reduced embedding matrix with kept tokens and cluster representatives
$all\_tokens \leftarrow tokens\_to\_keep \cup centers$
$token\_map \leftarrow \{old : new \mid new, old \in enumerate(all\_tokens)\}$
$reduced\_embeddings \leftarrow embeddings[all\_tokens]$
$M'.embedding\_layer.weight \leftarrow reduced\_embeddings$
return M', $token\_map$, oov_map

3.3 Out-of-Vocabulary Token Handling

A key contribution of this approach is the handling of out-of-vocabulary (OOV) tokens. Rather than mapping all pruned tokens to a single UNK token, the semantic properties of the embedding space are utilized to create meaningful representations for OOV tokens.

The clustering process organizes removed tokens into semantic groups, with each group represented by the token closest to the cluster centroid. When the model encounters an OOV token during inference, it maps to the appropriate cluster representative, preserving semantic properties. This approach retains more information than traditional UNK token methods.

3.4 Implementation Details

The implementation requires no additional fine-tuning after vocabulary pruning, improving efficiency compared to approaches that require retraining. The method comprises three technical components:

1. A token extraction and importance calculation module interfacing with downstream task datasets
2. An embedding space K-means clustering algorithm creating semantic maps for OOV tokens
3. A hybrid data collator handling tokenization and OOV mappings during model inference

Four pruning mechanisms are supported:

- Frequency-based pruning (retaining most frequent tokens)
- Importance-based pruning using non-normalized TF-IDF
- Importance-based pruning using L1-normalized TF-IDF
- Importance-based pruning using L2-normalized TF-IDF (default)

Experimental results indicate that TF-IDF variants generally outperform pure frequency-based approaches, with L2-normalized TF-IDF providing the optimal balance between task-specific vocabulary retention and general language understanding.

4 Experiments

Table 1: Comparison of vocabulary pruning techniques on GLUE benchmark tasks. Results show accuracy on the development set with varying pruning methods. Best results for each task are highlighted in **bold**.

Method	SST-2	MRPC	MNLI	CoLA	STS-B	QQP	QNLI	RTE	AVG
Baseline									
ModernBERT (Full)	0.951	0.856	0.881	0.586	0.917	0.917	0.916	0.598	0.828
Vocabulary Pruning									
Parameter Reduction	18.85%	18.56%	6.74%	22.61%	18.76%	4.79%	6.42%	17.06%	
Train Tokens Only	0.950	0.831	0.883	0.509	0.917	0.917	0.915	0.598	0.815
Parameter Reduction	20.25%	20.02%	10.57%	23.27%	20.18%	9.01%	10.31%	18.83%	
Random Selection	0.911	0.798	0.832	0.470	0.845	0.902	0.895	0.522	0.772
Clustering-Based	0.899	0.714	0.712	0.000	0.786	0.875	0.836	0.510	0.667
Frequency-Based	0.943	0.812	0.880	0.467	**0.905**	**0.916**	**0.920**	0.542	0.798
Attention-Based	**0.947**	**0.864**	0.864	**0.589**	0.885	0.912	0.912	0.550	0.803
TF-IDF Based	0.933	0.807	**0.875**	0.520	0.901	0.900	0.917	**0.606**	**0.807**
Freq + OOV	**0.938**	0.828	**0.881**	0.540	0.906	0.916	0.918	0.538	0.808
TF-IDF + OOV	0.923	0.834	0.878	0.615	0.903	0.906	0.919	0.554	**0.817**

Table 2: Average performance change across GLUE tasks with different pruning methods.

Method	Avg. Param. Reduction	Avg. Performance Drop
Train Tokens Only	14.22%	1.25%
Random Selection	16.44%	5.58%
Clustering-Based	16.44%	16.24%
Frequency-Based	16.44%	2.32%
Attention-Based	15.92%	1.99%
TF-IDF Based	16.44%	2.04%
Frequency + OOV	16.40%	1.82%
TF-IDF + OOV	16.40%	0.96%