CSE 256 PA2: Transformer Blocks

Sergi Marsol Torrent, PID A69042331

February 2026

1 Introduction

Transformer models are central to modern natural language processing due to their ability to capture complex contextual dependencies via self-attention. This report examines transformer architectures applied to two tasks: speech classification and language modeling.

The first task is a three-way classification problem, predicting which of three politicians delivered a given speech segment from tokenized transcripts. This evaluates the encoder's ability to generate contextual representations that support accurate classification. The second task is autoregressive language modeling, where the decoder predicts the next token in a sequence based on prior context, assessing its capacity to capture syntactic and semantic dependencies.

In addition to implementing standard transformer encoders and decoders from scratch, we explore architectural modifications to improve language modeling performance. Specifically, we examine sparse attention mechanisms and alternative positional encoding methods, such as ALiBi, to enhance efficiency, reduce overfitting, and encourage the model to focus on relevant local and long-range dependencies.

2 Part 1: Encoder with Classifier

2.1 Encoder Architecture

A transformer encoder was implemented from scratch for the three-way speech segment classification task, without using built-in PyTorch transformer modules. The encoder consists of four identical layers operating on tokenized input sequences of maximum length 32.

Each token is mapped to a learnable embedding of dimension $d_{\text{model}} = 64$, to which learnable absolute positional embeddings are added. Each encoder layer has two sublayers: a multi-head self-attention mechanism with two heads, and a position-wise feedforward network with a hidden dimension of $d_{\text{ff}} = 256$. Residual connections and layer normalization follow both sublayers. The encoder outputs contextualized token embeddings, which are mean-pooled across the sequence (excluding padded positions) to form a fixed-length representation for classification.

The encoder has 857,208 learnable parameters.

2.2 Classifier Architecture

A feedforward classifier predicts speaker identity from the pooled encoder output. The 64-dimensional representation is mapped to a hidden layer of size 100 with ReLU activation, then projected to an output of dimension 3 corresponding to the three politician classes. The classifier produces logits used with cross-entropy loss during training and contains 6,803 learnable parameters.

2.3 Training Details

The encoder and classifier were trained jointly in an end-to-end manner using the Adam optimizer with a learning rate of 10^{-3}. Training minimized cross-entropy loss over tokenized speech sequences of maximum length 32, with a batch size of 16. The model was trained for 15 epochs, with the pooled encoder outputs producing logits for the three speaker classes.

2.4 Attention Sanity Check

Encoder attention maps were examined to verify the correct implementation of multi-head self-attention. The attention matrices satisfied the normalization constraint, with each row summing to one after the softmax.

Figure 1 shows self-attention heatmaps from Layer 1 Head 1 and Layer 4 Head 1 for the input sentence *"But new threats also require new ¡unk¿."*. The shallow layer exhibits diffuse patterns, while the deeper layer shows more structured interactions between semantically related tokens. Notably, the word "threats" receives the most attention from other tokens, highlighting its relevance in context. This progression indicates that deeper encoder layers capture increasingly refined contextual dependencies across the sequence.

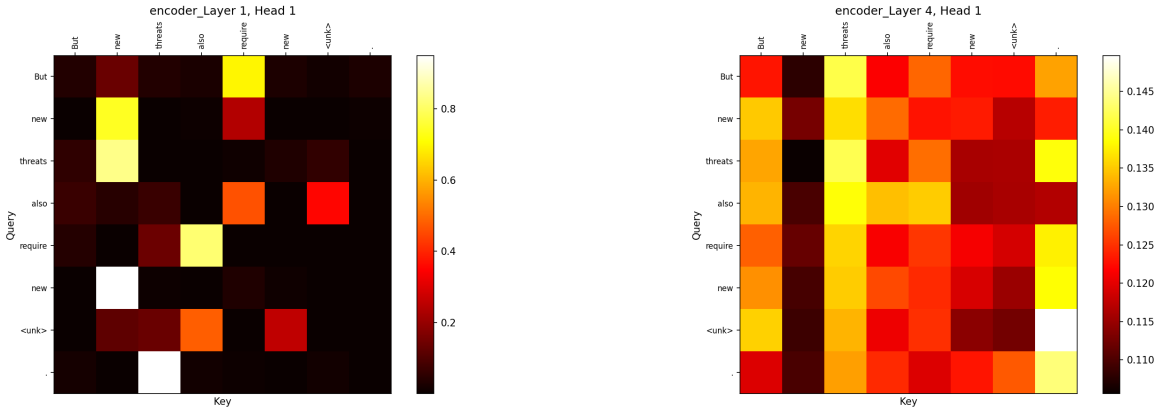

Figure 1: Encoder self-attention heatmaps for the sentence "But new threats also require new 'unk'." from Layer 1 Head 1 (left) and Layer 4 Head 1 (right).

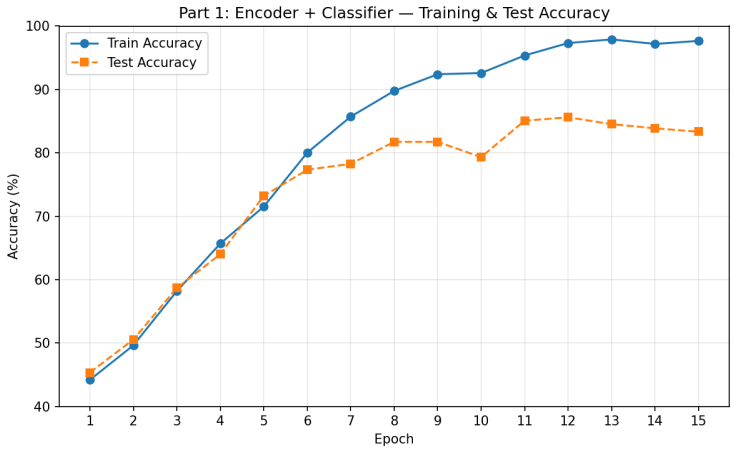

Figure 2: Training and test classification accuracy across training epochs for the encoder-based classifier.

2.5 Results

Figure 2 shows the training and test classification accuracy as a function of training epoch. Accuracy improves steadily during the early stages of training, indicating that the encoder learns meaningful discriminative representations for the speech classification task. Performance peaks around epoch 12, at around 0.86, after which a slight decline in test accuracy is observed, suggesting mild overfitting in later epochs in the absence of explicit regularization.

3 Part 2: Decoder Language Model

3.1 Decoder Architecture

A GPT-like transformer decoder was implemented from scratch for the autoregressive language modeling task. The decoder consists of four identical layers operating on tokenized sequences of length 32.

Each layer includes a masked multi-head self-attention mechanism with two heads, followed by a position-wise feedforward network. Queries, keys, and values are computed via learned linear projections of the input embeddings. To enforce the autoregressive constraint, a causal mask is applied as an upper-triangular matrix: token position i can attend only to positions $\leq i$, with masked entries set to $-\infty$ prior to softmax.

The feedforward sublayer uses two linear transformations with ReLU activation, mapping from $d_{\text{model}} = 64$ to a hidden dimension of 100 and back. Layer normalization is applied after all decoder blocks, and a final linear projection maps to the vocabulary for next-token prediction. The decoder contains 864,011 learnable parameters.

3.2 Training Setup

The decoder was trained with an autoregressive next-token prediction objective, using cross-entropy loss between predicted distributions and ground-truth tokens. Model parameters were optimized with the Adam optimizer at a learning rate of 10^{-3}. Training ran for 500 iterations with a batch size of 16 and block size 32, processing roughly 256,000 tokens in total.

3.3 Attention Sanity Check

Attention maps were examined to confirm correct causal masking. All upper-triangular entries in the attention matrices were zero, ensuring tokens could not access future positions. Figure 3 shows masked self-attention heatmaps for Layer 1 Head 1 and Layer 4 Head 1 of the sentence *"But new threats also require new ¡unk¿."*. The strictly lower-triangular pattern confirms proper autoregressive behavior. Unlike the encoder, differences between shallow and deep layers are less pronounced, suggesting that deeper layers refine token representations without drastically changing attention patterns.

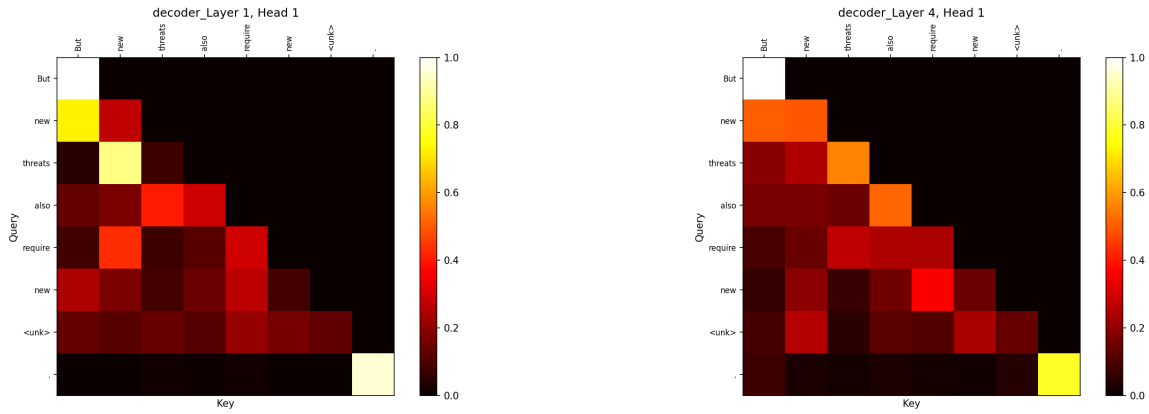

Figure 3: Decoder masked self-attention heatmaps for "But new threats also require new ¡unk¿." from Layer 1 Head 1 (left) and Layer 4 Head 1 (right).

3.4 Results

Figure 4 shows the training perplexity across 500 iterations. Perplexity decreases steadily, indicating the decoder learns to predict the next token using only past context. The final training perplexity stabilizes around 201, confirming the causal mask prevents access to future tokens.

On held-out test sets, the model achieves perplexities of 406.80 for Obama, 492.74 for W. Bush, and 432.61 for H. Bush. The higher value for W. Bush suggests his speeches diverge most from the training distribution, likely due to distinct vocabulary and rhetorical style, particularly on post-9/11 security and foreign-policy topics. Obama's speeches are captured most effectively, while H. Bush falls in between, reflecting his earlier-era style that partially overlaps with W. Bush but differs in formality and content.

Overall, these results indicate that the decoder, with 864,011 parameters, effectively models next-token distributions while highlighting how speaker style and content influence generalization performance.

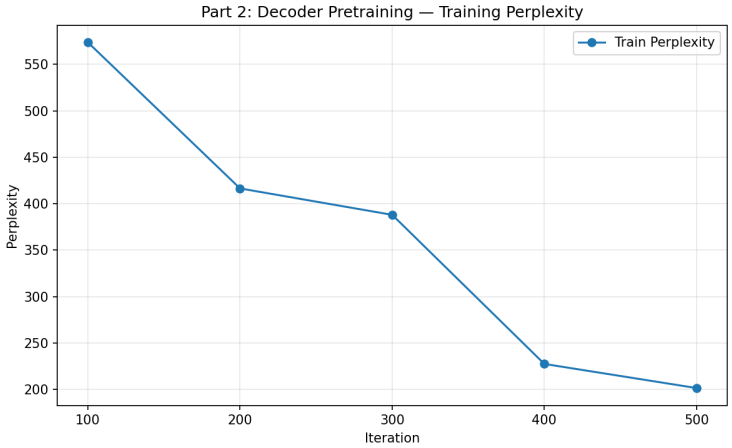

Figure 4: Training perplexity as a function of training iteration for the decoder language model.

4 Part 3: Architectural Exploration

4.1 Motivation

Transformer architectures rely on full self-attention and learned positional embeddings to model contextual dependencies between tokens. However, full causal attention incurs $\mathcal{O}(n^2)$ computational complexity and distributes attention across all preceding tokens, which may include irrelevant long-range dependencies.

Sparse attention mechanisms aim to improve efficiency and generalization by restricting attention to a subset of tokens, encouraging the model to focus on more informative local context. Additionally, alternative positional encoding schemes such as ALiBi introduce inductive biases that emphasize the relative importance of nearby tokens without relying on learned positional embeddings. These modifications can improve performance in low-data regimes by reducing overfitting and encouraging locality-aware representations.

4.2 Explored Architectures

Three architectural modifications were explored:

- **Local Window Sparse Attention:** Each token attends only to the W most recent preceding tokens, restricting the attention span to a fixed sliding window.
- **Block Sparse Attention:** The input sequence is partitioned into fixed-size blocks, and each token attends only within its own block and the immediately preceding block.
- **ALiBi Positional Encoding:** Learned positional embeddings are replaced with per-head linear biases added directly to the attention scores based on relative token distance.

All experiments were conducted using identical hyperparameters ($d_{\text{model}} = 64$, $n_{\text{head}} = 2$, $n_{\text{layer}} = 4$, block size $= 32$) and trained for 500 iterations. The total number of learnable parameters remained unchanged at 864,011 for all configurations, ensuring a fair comparison between architectural variants.

4.3 Results

Configuration	Obama	W.Bush	H.Bush	Avg
Full Attention	395.51	488.96	430.65	438.37
Window (W=5)	377.81	454.91	399.03	410.58
Window (W=3)	371.15	469.33	405.88	415.45
Block Sparse	393.90	475.48	425.53	431.64
ALiBi	354.39	450.14	385.70	396.74

Table 1: Test perplexity comparison across architectural configurations.

Table 1 reports the best test perplexity obtained for each architectural configuration across the three evaluation datasets. All sparse attention variants outperform the baseline full-attention decoder, with ALiBi achieving the lowest perplexity on all three test sets.

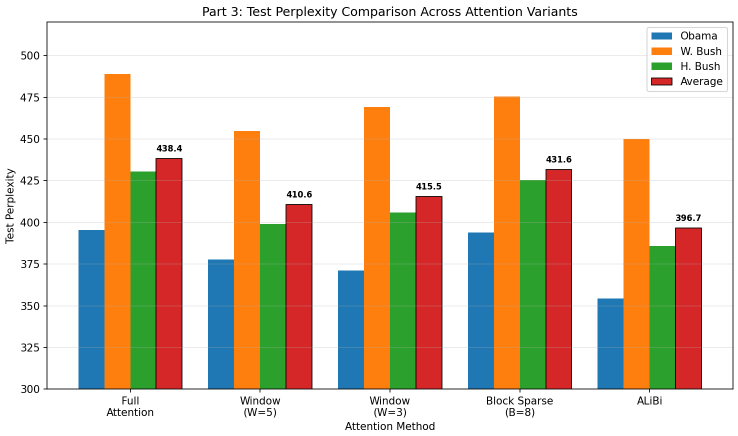

Figure 5: Test perplexity comparison across architectural variants and their average performance.

As shown in Figure 5, ALiBi yields the lowest average perplexity among all evaluated configurations, improving performance substantially relative to full attention. Local window sparse attention also provides a notable reduction in perplexity, suggesting that restricting attention to nearby tokens helps the model capture relevant short-range dependencies while reducing overfitting. In contrast, block sparse attention results in only a modest improvement, likely due to the rigid block boundaries limiting access to useful context near block edges.

The superior performance of ALiBi can be attributed to its soft distance-based inductive bias. Unlike sparse attention mechanisms that hard-mask distant tokens, ALiBi allows all tokens to contribute to the attention computation while gradually reducing the influence of tokens that are farther away. This enables the model to retain useful long-range context when necessary while still prioritizing local dependencies. Additionally, the use of head-specific linear biases encourages multi-scale attention behavior, allowing different heads to focus on varying contextual ranges. These properties likely contribute to improved generalization and lower perplexity across all evaluation datasets.

4.4 Discussion

The observed performance gains suggest that restricting attention to local context can act as an effective form of regularization, reducing reliance on long-range dependencies. However, hard sparsity patterns like local windowing or block partitioning may prevent the model from leveraging global information when needed.

In contrast, ALiBi introduces a soft positional bias directly into the attention scores, allowing the model to prioritize nearby tokens while still retaining access to the full sequence context. This flexibility likely contributes to its consistently superior performance across all datasets, as seen in Table 1 and Figure 5.

5 Conclusion

In this report, I implemented transformer-based encoder and decoder architectures from scratch for speech classification and language modeling. The encoder, trained with a three-way classifier, achieved steadily improving accuracy, while the decoder showed decreasing training perplexity, confirming effective next-token prediction under causal constraints.

Architectural exploration demonstrated that sparse attention, especially local windowing, improves generalization by focusing on relevant recent tokens, and ALiBi positional encoding achieves the lowest test perplexity by introducing soft distance-based biases that retain long-range context. These findings underscore the value of balancing local and global information and show that careful architectural design and targeted modifications can enhance transformer performance for sequential data tasks.

Annex: Use of Generative AI

Generative AI tools were utilized during the preparation of this project. Specifically, AI assistance was employed to clean up and format code, add clear comments, and improve readability. Additionally, AI was used to refine the language, structure, and presentation of both the README and this report. It was not used to directly write the implementation or produce original code; all code logic, design choices, and experiments were developed independently.