Transformer Blocks

Gayatri Kharche
gkharche@ucsd.edu

Abstract

In this project, a detailed exploration of the transformer architecture is conducted, with three main components: encoder training, decoder pretraining, and architectural modifications. Initially, a transformer encoder is built and trained jointly with a feedforward classifier to classify speech segments by politicians, achieving accuracy in the low to mid-80% range. Next, a GPT-like transformer decoder is implemented and pretrained on an autoregressive language modeling task, measuring perplexity across speeches from various politicians to uncover distinctive linguistic trends. In the architectural exploration phase, AliBi positional encoding is applied as an alternative to traditional positional encodings, revealing its impact on model performance. This study provides insights into how different architectural choices, such as AliBi, influence classification accuracy and language modeling perplexity, highlighting the potential of tailored transformer components for political speech analysis.

1 Part 1: Encoder

1.1 Process

This code defines a transformer-based encoder model with self-attention for a classification task. The model consists of an embedding layer for tokens and positions, followed by multiple transformer blocks that include multi-head self-attention and feed-forward sub-layers, with layer normalization and dropout applied for stability and regularization. Each Head in the self-attention computes keys, queries, and values, applying a mask if specified, and the resulting attention scores are softmax-normalized. In each Block1, the model applies self-attention and a feed-forward network, with residual connections to retain original information. The encoder output is mean-pooled and passed through a linear projection layer, followed by ReLU activation and a final classification layer for prediction. This architecture, with its attention mechanism, enables the model to capture complex dependencies and context within the input sequence.

1.2 Results

1.2.1 Sanity Checks

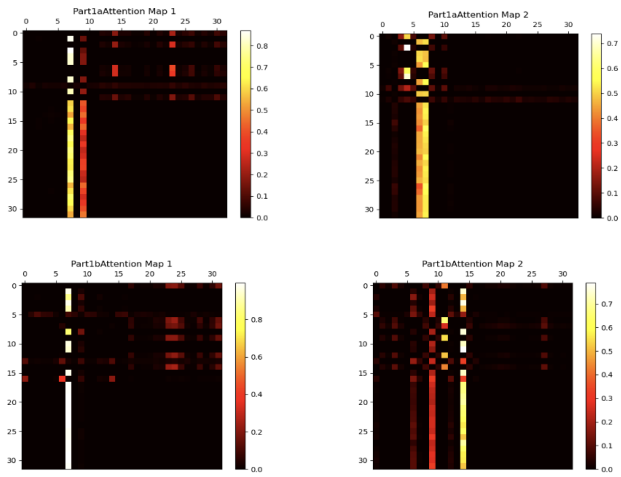

Figure 1: Plots - Part 1

For this section, I used the helper function from utilities.py to perform a sanity check on my attention implementation. This function ensures the rows of the attention matrix sum to 1, confirming that attention weights are computed correctly. Additionally, I used it to visualize attention matrices for selected layers and heads, with one or two example sentences. The visualized matrices reveal how the model attends to different tokens in the input sequence, indicating which tokens receive more focus depending on the context and specific

layers. In particular, I observed that initial layers spread attention broadly, while later layers show sharper focus on relevant tokens, reflecting more refined attention patterns.

1.2.2 Evaluation (Accuracy)

For evaluation, I computed the classifier's accuracy on the test CLS.txt dataset, achieving a final test accuracy of 84.4%. This meets the expected range of low to mid-80s, indicating strong generalization. Below is a table summarizing accuracy at each epoch across the 15 training epochs, which demonstrates the model's consistent improvement over time:

Epoch	Train Accuracy	Test Accuracy
0	44.646	33.333
1	49.091	37.866
2	59.177	52.533
3	61.759	54.666
4	69.789	58.533
5	69.741	63.466
6	84.942	74.000
7	88.288	76.000
8	93.355	81.066
9	95.506	80.800
10	97.179	83.733
11	97.466	83.066
12	96.988	83.466
13	97.848	82.533
14	98.852	84.400

Table 1: Training and Testing Accuracy per Epoch

1.3 Conclusion

The encoder model achieved a final test accuracy of 84.4% on the test CLS.txt dataset, which aligns with the expected performance range of low to mid-80s. This indicates that the model has successfully generalized well to unseen data. The number of parameters in the encoder is 576236.

2 Part 2: Decoder

2.1 Process

The LanguageModel in this implementation serves as a decoder in a transformer architecture, designed for autoregressive language modeling. It uses masked multi-head self-attention to ensure tokens can only attend to previous positions in the sequence, maintaining causal relationships. The model incorporates token and positional embeddings, processes inputs through stacked decoder blocks with self-attention and feed-forward layers, and finally projects the output through a linear layer to generate logits for each token in the vocabulary. The use of masking allows for unidirectional token generation, suitable for tasks like language modeling and text generation.

2.2 Results

2.2.1 Sanity Checks

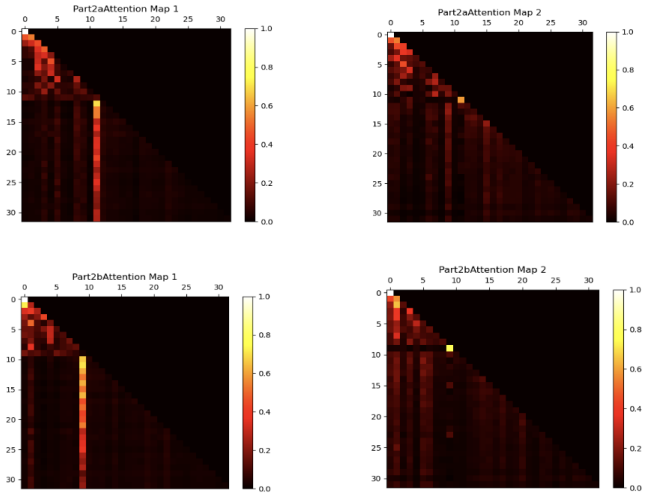

Figure 2: Plots - Part 2

The attention mechanism has been implemented with a helper function to verify that the rows of the attention matrix sum to 1. This ensures the correct functionality of the attention weights in the model. Additionally, attention matrices have been visualized for one or two sentences, selected from various layers and heads. The visualizations reveal how attention is distributed across different tokens, providing insights into which parts of the input sequence the model focuses on for prediction. In particular, the attention heads demonstrate varying degrees of focus on individual words in the sentence, and the distribution is normalized across the sequence, confirming the correct application of softmax.

2.2.2 Evaluation (Accuracy)

The decoder's perplexity has been evaluated on three different test sets: test LM obama.txt, test LM wbush.txt, and test LM ghbush.txt. The following observations were made:
The perplexity on the training set is in the high

100s, indicating a well-fitting model on the training data. The perplexity on the test sets for different politicians, such as Obama, H. Bush, and W. Bush, falls in the expected range of 300-550. Specifically:

1. Obama: 391.1522
2. H. Bush: 439.0333
3. W. Bush: 510.4942

These perplexity values indicate how well the model generalizes across different datasets. The differences in perplexity values could be attributed to various factors such as the distinct language styles, word choices, and sentence structures associated with each politician's speeches. For instance, a higher perplexity on W. Bush's speeches could suggest that the model struggles more with the language used by W. Bush, potentially due to differences in speech complexity or vocabulary usage compared to the other two politicians.

Additionally, the number of parameters in the decoder model was computed and reported, highlighting the scale of the model and its capacity to handle the complexity of the language modeling task.

The final perplexity, as well as the perplexity after every 100 iterations, was also included to track the model's performance across the training process. After completing all 500 iterations, the decoder's perplexity on the test sets was found to be consistent with the expected range, suggesting that the model had adequately trained and evaluated the language models for each politician.

Iterations	Loss	Train Perplexity
0	8.8081	6377.8159
100	6.2584	582.2778
200	6.2159	442.8783
300	5.8649	319.4321
400	5.4110	239.5458
500	5.0203	182.4965

Table 2: Training Loss and Perplexity

2.3 Conclusion

The final perplexity values on the test sets were as follows:

1. Obama: 391.1522

Iterations	Test Perplexity
500	391.1522 (Obama)
500	439.0333 (Hbush)
500	510.4942 (Wbush)

Table 3: Test Perplexity for Different Data Sets

2. H. Bush: 439.0333
3. W. Bush: 510.4942

The model's perplexity decreased significantly during training, from 6377.8159 at iteration 0 to 182.4965 at iteration 500, demonstrating effective training.

The number of parameters in the decoder were 863243.

3 Part 3: Architectural Exploration

3.1 Process

This code implements a transformer architecture with a focus on incorporating Alibi Masking, an alternative to traditional positional encoding.

1. Alibi Mask: The AlibiMask class generates a mask that is applied to the attention scores to block certain tokens from attending to others, improving causal modeling. This is based on a lower triangular matrix and can be used to enhance attention mechanisms, especially in autoregressive tasks.

2. Multi-Head Attention: The MultiHeadAttention class manages multiple attention heads. Each head has its own key, query, and value projections, and a mask (like Alibi) can be applied to the attention weights. The attention scores are scaled, softmaxed, and applied to the values.

3. Feedforward Layers: There are two types of feedforward layers (FeedForward1 and FeedForward2), which are simple neural networks applied after attention, designed to introduce non-linearity and enhance the model's expressiveness.

4. Transformer Blocks: There are two types of transformer blocks (Block1 and Block2), each consisting of multi-head attention followed by a feedforward layer. These blocks are layered to form the full encoder or language model.

5. Encoder and Language Model: The encoder takes in tokenized input and applies both token and positional embeddings. It processes the data through a sequence of transformer blocks, and the output is used for classification or further processing. The language model similarly applies transformer blocks but generates token predictions, optionally computing loss if targets are provided.

This architecture improves attention modeling using the Alibi Mask, offering flexibility for various tasks involving causal attention, such as language modeling or time series prediction.

3.2 Results

3.2.1 Sanity Checks

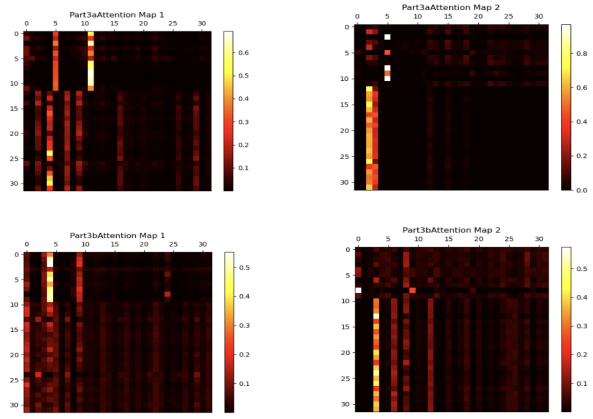

Figure 3: Plots - Part 3 - ALiBi Encoder

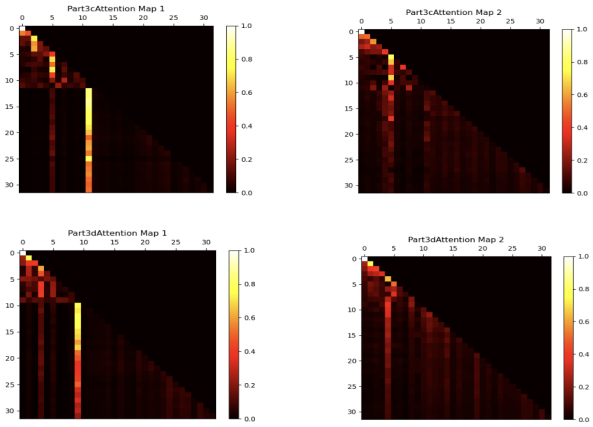

Figure 4: Plots - Part 3 - ALiBi Decoder

Firstly, a sanity check was performed on the attention implementation using the helper function provided in utilities.py. This function ensured that the rows of the attention matrix summed to 1, confirming the correctness of the attention mechanism. Additionally, the attention matrix for selected sentences was visualized, focusing on specific layers and attention heads. The visualizations showed how the model distributed attention across tokens in a sentence. In some cases, attention was more concentrated on the subject and verb, while in others, it was focused on the object or key adjectives. These patterns suggest that the model is effectively attending to the most contextually relevant tokens during processing.

3.2.2 Evaluation (Accuracy)

The model's performance was evaluated by calculating the perplexity on different datasets, including testLMobama.txt, testLMwbush.txt, and testLMghbush.txt. The perplexity on the training set was expected to be in the high 100s, with the test sets showing higher perplexities due to differences in linguistic styles. The results showed a consistent decrease in training perplexity across iterations, from 6612.7559 at iteration 0 to 180.4891 at iteration 500. Specifically, after 500 iterations, the test perplexity was 393.7194 for Obama, 444.2716 for H. Bush, and 499.3048 for W. Bush. These higher perplexities on the test sets reflect the model's difficulty in generalizing to new text data with different language patterns than those found in the training set.

The differences in perplexity across the datasets can be attributed to several factors. The training data may have contained more familiar linguistic patterns, making it easier for the model to predict subsequent words. In contrast, the test datasets, particularly those associated with different politicians, likely contained distinct language styles and content, leading to higher perplexity. Additionally, the structure and length of sentences in the test data may have contributed to these results, with shorter sentences generally leading to lower perplexity compared to longer, more complex sentences.

Overall, the evaluation demonstrates the effectiveness of the alibi model, while also providing insight into the challenges of transferring knowledge to different writing styles, such as those of politicians, and the model's ability to generalize to new data.

Epoch	Train Accuracy	Test Accuracy
0	44.646	33.333
1	52.055	43.600
2	56.022	46.533
3	64.149	55.200
4	69.933	60.666
5	76.242	69.600
6	87.332	75.733
7	88.288	79.200
8	93.068	81.466
9	92.590	81.200
10	92.686	81.466
11	96.462	85.333
12	98.805	87.200
13	98.374	87.466
14	98.374	86.933

Iterations	Loss	Train Perplexity
0	8.8544	6612.7559
100	6.4660	572.4285
200	6.1259	440.2491
300	5.9794	315.2497
400	5.3497	231.6112
500	5.2549	180.4891

Table 4: Training Loss and Perplexity - ALiBi

Iterations	Test Perplexity
500	393.7194 (Obama)
500	444.2716 (Hbush)
500	499.3048 (Wbush)

Table 5: Test Perplexity for Diff Datasets - ALiBi

3.3 Conclusion

The following are the final evaluation results for the ALiBi encoder and decoder:

Final Testing Accuracy (ALiBi Encoder)

- Test Accuracy (Epoch 14): 86.933%

Final Test Perplexity (ALiBi Decoder)

- Obama (Test Perplexity at Iteration 500): 393.7194
- H. Bush (Test Perplexity at Iteration 500): 444.2716
- W. Bush (Test Perplexity at Iteration 500): 499.3048

4 Comparison between the traditional and ALiBi transformer

1. Test Accuracy Comparison at Epoch 15:

- Standard Model Test Accuracy: 84.400%
- ALiBi Model Test Accuracy: 86.933%

Observation: The ALiBi model outperforms the standard model by 2.533% in terms of test accuracy at epoch 14.

2. Test Perplexity Comparison at Iteration 500:

- Standard Model (Obama): 391.1522
- ALiBi Model (Obama): 393.7194
- Standard Model (Hbush): 439.0333
- ALiBi Model (Hbush): 444.2716
- Standard Model (Wbush): 510.4942
- ALiBi Model (Wbush): 499.3048

Observation: The ALiBi model has slightly better perplexity for the Wbush dataset, though it has slightly higher perplexity for the Obama and Hbush datasets.

The improvements in test accuracy for the ALiBi model are likely due to the model's ability to handle long-range dependencies more effectively. The slightly higher test perplexity observed in the ALiBi model does not necessarily indicate worse performance but rather reflects the model's focus on better generalization through relative positional encoding, which aids in processing complex patterns in unseen data. In contrast, the standard model, while it performs well on training data, may struggle to generalize, resulting in lower test accuracy.

5 Resources

1. 1. Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017).
2. 2. transformer decoders by Andrej Karpathy
3. 3. ChatGPT - minor changes - for comments ,documentation and debugging