

CSE 256 PA2

Transformer Blocks Report

Chuhan Wang

0. Introduction

This report outlines the construction of Transformer blocks for both encoder and decoder segments, specifically focusing on the integration of the attention mechanism. Additionally, it examines the implementation of different positional embeddings within the framework of Transformer. The objective was to assess the impact of these modifications on tasks such as speaker identification from speech segments and next-word prediction in textual sequences.

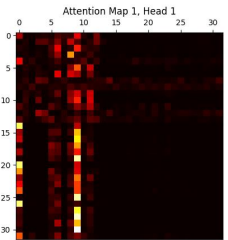
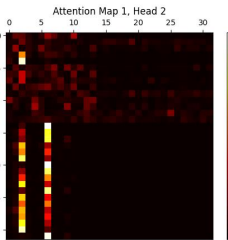
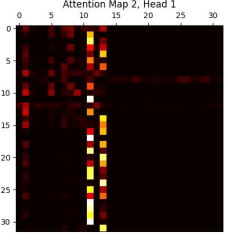
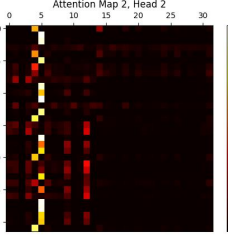
1. Encoder Trained With Classifier

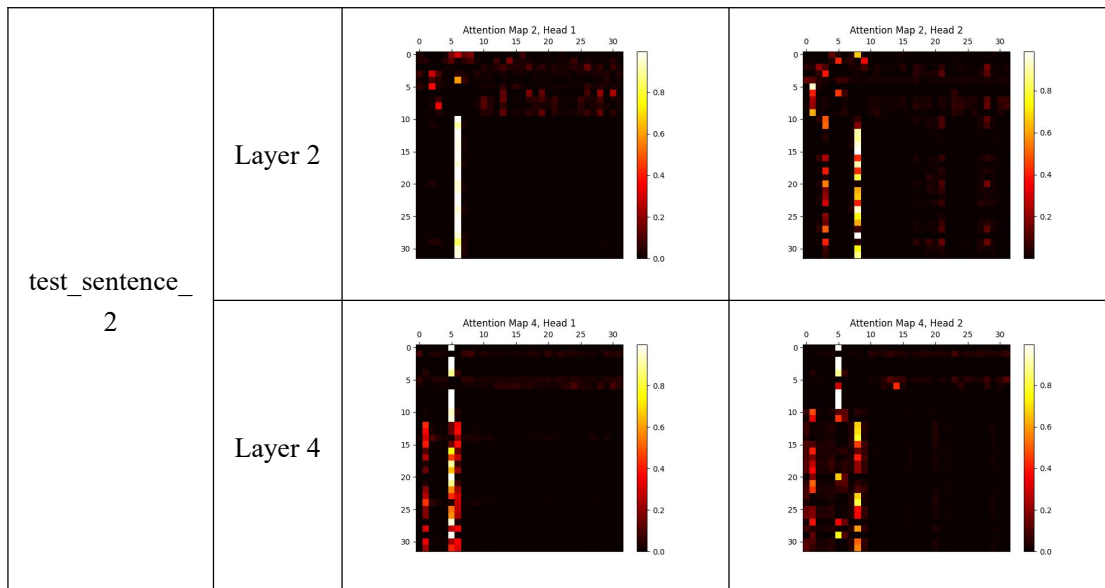
Introduction

The first section built and trained a Transformer encoder. This setup was used to identify speakers from speech segments without any pre-training. This part discusses the training process and the overall accuracy of our model under default configurations.

(a) Sanity Checks

- The rows of the attention matrix of each head sum to 1.
- Choose 2 test sentences from the file `'test_CLS.tsv'`.
test_sentence_1 = "But I stand before you tonight because all across America something is stirring."
test_sentence_2 = "I will rebuild our military to meet future conflicts."
- Attention matrix visualization. A few examples are shown below.

| Test sentence | | Head 1 | Head 2 |
|-----------------|---------|---|---|
| test_sentence_1 | Layer 1 |  |  |
| | Layer 2 |  |  |

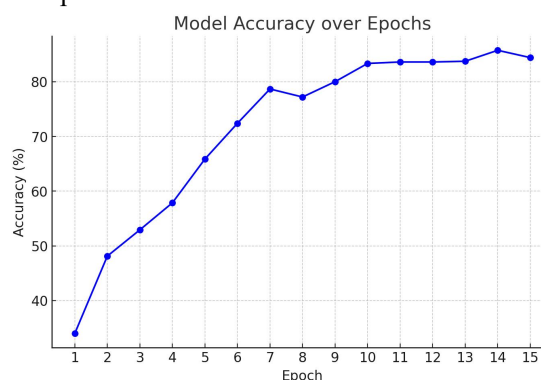


- The figures presented show that attention maps vary according to the text being processed, as well as across different heads within the same text. Additionally, the attention maps corresponding to a single head change across different layers. This variability highlights the nuanced behaviors of attention mechanisms, adapting dynamically to the input text's context as well as the depth of the processing layer within the model.

- In *test_sentence_1*, the word 'America' receives significant attention in head 1 of layer 1, while the words 'stand,' 'before,' and 'you' show strong mutual attention. In *test_sentence_2*, the word 'military' is closely attended to alongside 'meet', and both instances of the word 'to' as well as 'conflicts' garner considerable attention. These observations suggest that the model's attention mechanism effectively focuses on key words and phrases that are likely central to the meaning of the sentence.

(b) Evaluation

- number of parameters in my encoder: 569,280
- Test accuracy after each epoch:



Specifically,

| Epoch | Accuracy (%) | Epoch | Accuracy (%) |
|-------|-------------------|-------|-------------------|
| 1 | 34.0 | 9 | 80.0 |
| 2 | 48.13333333333333 | 10 | 83.33333333333333 |
| 3 | 52.93333333333333 | 11 | 83.6 |
| 4 | 57.86666666666667 | 12 | 83.6 |

| | | | |
|---|-------------------|----|-------------------|
| 5 | 65.86666666666666 | 13 | 83.73333333333333 |
| 6 | 72.4 | 14 | 85.73333333333333 |
| 7 | 78.66666666666667 | 15 | 84.4 |
| 8 | 77.2 | | |

- The highest accuracy achieved over the 15 epochs is 85.73333333333333%, with the final accuracy settling at 84.4%. A noticeable pattern is that the accuracy experiences periodic declines after several epochs. This issue may be attributed to the settings of the learning rate. By reducing the learning rate from 1e-3 to either 1e-4 or 2e-4, the situation showed improvement.

2. Pretraining Decoder Language Model

Introduction

The second part covers work on a Transformer decoder designed for language modeling. This includes the implementation of masked self-attention to predict the next words in a text and how to measure the model's effectiveness using perplexity across different speech datasets.

(a) Implementation of the Mask Mechanism

Step 1: To create the mask, use `torch.tril` on the base attention matrix from Part 1 to keep the lower triangular part as 1 and set the rest to 0. This operation defines the mask area.

Specific implementation:

```
mask = torch.tril(torch.ones((seq_len, seq_len), device=x.device)).unsqueeze(0).unsqueeze(0)
```

Step 2: Before computing the product of QK^T (the attention matrix) and V (values), adjust the weights in the attention matrix where it is zero to negative infinity ($-\infty$). This ensures that the masked areas are completely excluded when calculating the weighted values.

Specific implementation:

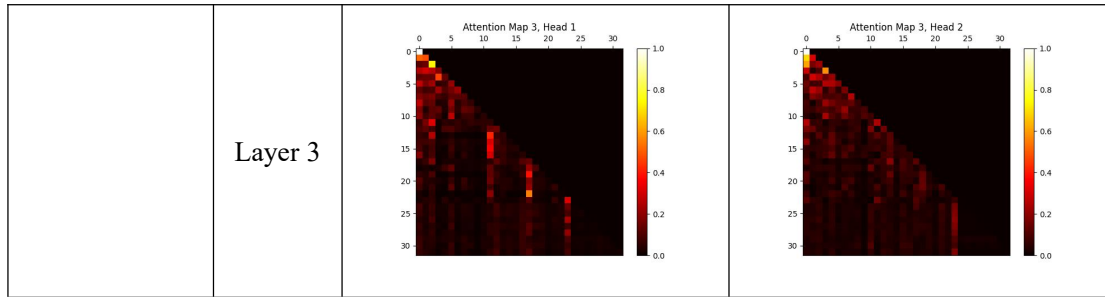
```
scores = scores.masked_fill(mask == 0, float('-inf'))
```

(b) Sanity Checks

- The rows of the attention matrix of each head sum to 1.
- Choose a test sentence from the file `'test_LM_wbush.txt'`.

```
test_sentence = "We cannot put our faith in the word of tyrants, who solemnly sign non-proliferation treaties, and then systemically break them."
```
- Attention matrix visualization. A few examples are shown below.

| Test sentence | | Head 1 | Head 2 |
|---------------|---------|--------|--------|
| test_sentence | Layer 1 | | |



- Same as Part 1, the attention maps vary according to the text being processed, as well as across different heads within the same text. Additionally, the attention maps corresponding to a single head change across different layers.
- The masked self-attention mechanism ensures that each input token only pays attention to itself and the tokens that precede it, which aligns with the logic required for language generation tasks. This design prevents tokens from attending to future tokens in the sequence, maintaining the sequential integrity necessary for accurately predicting the next word in text generation models.
- In the initial layers, tokens focus mainly on themselves and nearby tokens, capturing local context essential for understanding syntax. As the model progresses through deeper layers, this focus shifts from immediate to broader contexts, indicating a transition from detailed linguistic features to global semantic understanding within the network.

(c) Evaluation

- number of parameters in my decoder: 943,355
- Training perplexity after every 100 iterations:

| Iteration | Perplexity |
|-----------|-------------------|
| 0 | 5808.12548828125 |
| 100 | 590.0912475585938 |
| 200 | 390.12841796875 |
| 300 | 266.2806701660156 |
| 400 | 198.7044677734375 |

- Evaluation on test sets:

| Test sets | Perplexity |
|-------------------|--------------------|
| test_LM_obama.txt | 362.4099426269531 |
| test_LM_wbush.txt | 447.51483154296875 |
| test_LM_hbush.txt | 392.5148010253906 |

- The variation among the perplexities could stem from several factors:
 1. Each politician's unique speaking style, vocabulary use, and sentence structure directly influence the model's predictive efficiency, resulting in varied perplexity scores.
 2. How well the training data represents the linguistic features of each politician's speeches affects perplexity. Better alignment between training data and a politician's speech characteristics can lead to lower perplexity on related test data.
 3. The model's ability to generalize from seen training data to unseen test data can impact perplexity, influenced by potential overfitting or underfitting during the training process.

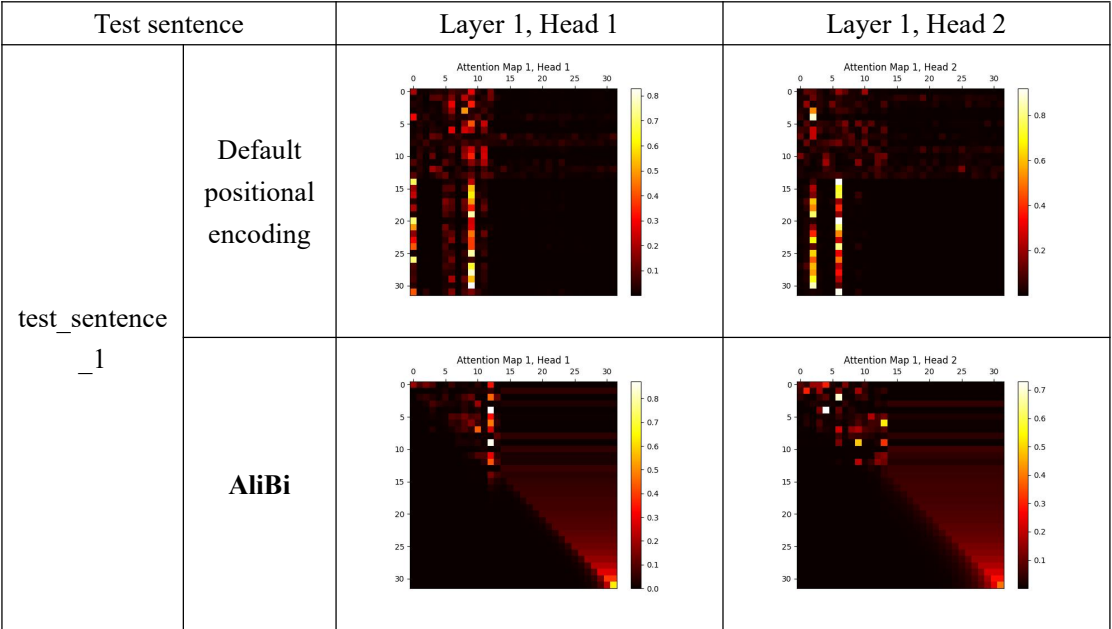
3. Architectural Exploration

Introduction

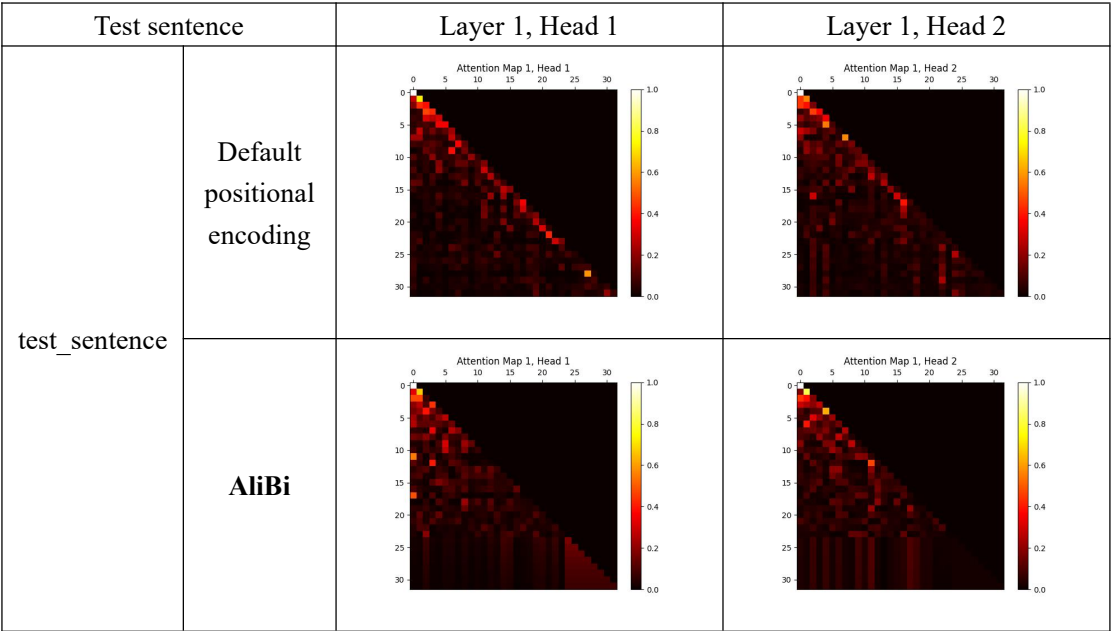
The final part looks into the experiments with modifying the Transformer architecture. I tried a **different positional encoding way -- AliBi** -- to see how it could make the model more efficient and adaptable to different types of data.

(a) Sanity Checks

Encoder:



Decoder:



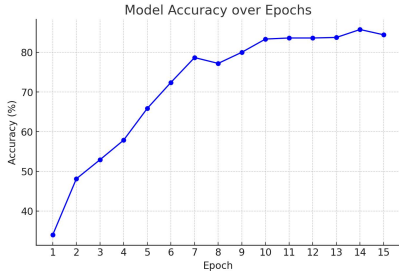
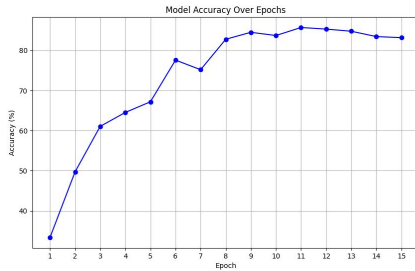
Findings: The default positional encoding displays scattered attention across the sequence, both along and away from the diagonal; whereas AliBi shows more focused attention around fewer tokens, especially along the diagonal, suggesting enhanced local context capture.

(b) Evaluation

Encoder:

- Test accuracy

| | | |
|--|-----------------------------|-------|
| | Default positional encoding | AliBi |
|--|-----------------------------|-------|

| | | |
|--------------------------------|---|--|
| test accuracy after each epoch |  |  |
| Highest accuracy | 85.73333333333333 | 85.73333333333333 |

Findings: The AliBi encoding converges more quickly and becomes notably stable in later stages, indicating more consistent performance across the sequence. This behavior may enhance the model's efficiency and reliability in processing language data.

Decoder:

- Training perplexity after every 100 iterations:

| Iteration | Default positional encoding | AliBi |
|-----------|-----------------------------|--------------------|
| 0 | 5808.12548828125 | 6107.494140625 |
| 100 | 590.0912475585938 | 454.1864929199219 |
| 200 | 390.12841796875 | 268.7591857910156 |
| 300 | 266.2806701660156 | 177.86029052734375 |
| 400 | 198.7044677734375 | 130.19393920898438 |

- Perplexity Evaluation on test sets:

| Test sets | Default positional encoding | AliBi |
|-------------------|-----------------------------|-------------------|
| test_LM_obama.txt | 362.4099426269531 | 342.8024597167969 |
| test_LM_wbush.txt | 447.51483154296875 | 432.9507141113281 |
| test_LM_hbush.txt | 392.5148010253906 | 375.8157958984375 |

Findings: The AliBi positional encoding demonstrates a clear advantage over the Default encoding in both training and testing phases. During training, AliBi shows a quicker decrease in perplexity, reaching significantly lower values at each measured interval. On test sets, AliBi consistently yields lower perplexity (15-20 lower than default) scores across all texts, indicating improved model generalization and efficiency in handling language tasks. Theref AliBi is more effective in capturing and utilizing the contextual information necessary for language modeling.

4. References

- [1] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. CoRR, abs/2004.05150, 2020.
- [2] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: decoding-enhanced bert with disentangled attention. In ICLR. OpenReview.net, 2021.
- [3] Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In ICLR, 2022.
- [4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NeurIPS, pages 5998–6008, 2017.