

Max Element In a List

Problem: problem 66 from neel's 200 concrete problem interpretability (beginner problem).

Task & dataset

Sequence = [7, 5, 11, 4, 1, 16, 11]

Str toks = ['7', '5', '11', '4', '1', 'SEP', '11']

Model

The model is attention-only, with 1 layer, and 2 attention heads per layer. It was trained with layernorm, weight decay, and an Adam optimizer with linearly decaying learning rate.

Average cross entropy loss: 0.006

Mean probability on correct label: 0.994

Median probability on correct label: 0.997

Min probability on correct label: 0.975

Hypothesis

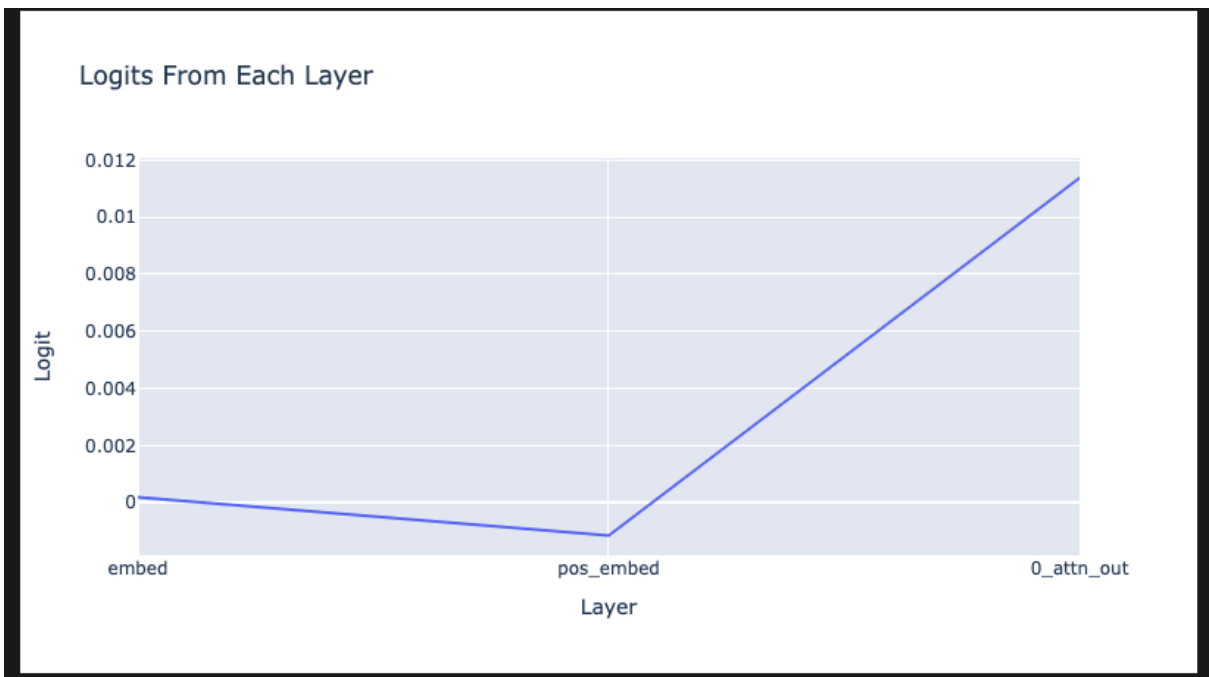
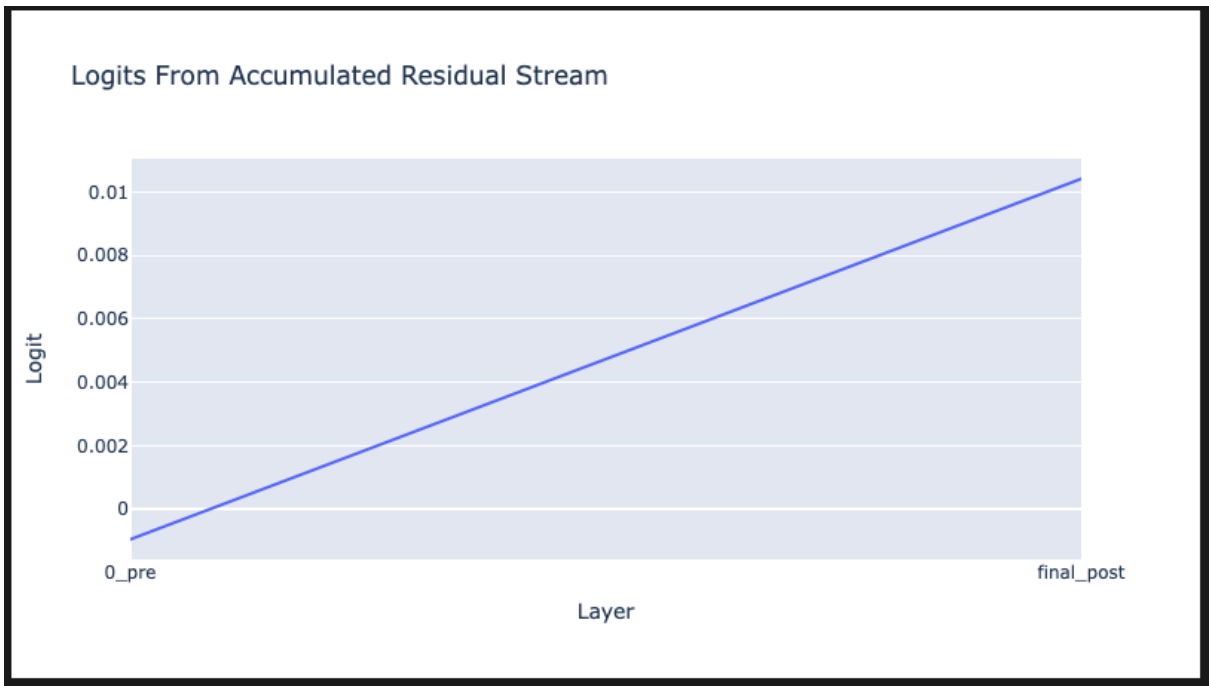
Embedding layer already stores some concept of the magnitude of each token.

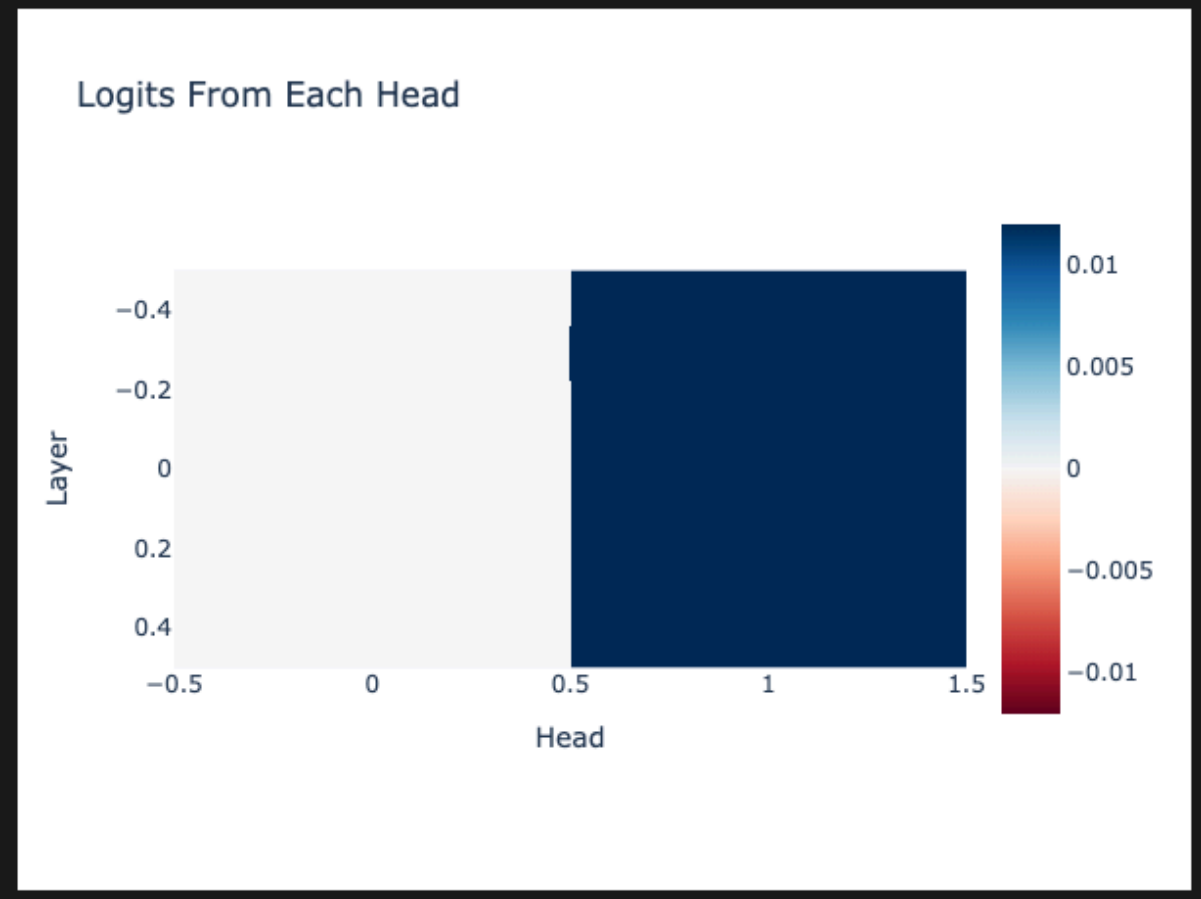
Curious to see if the two heads do something different, same thing, or one of them does nothing at all

I expect every destination token to attend to all the other tokens before it and sort of arrange them according to their magnitude?

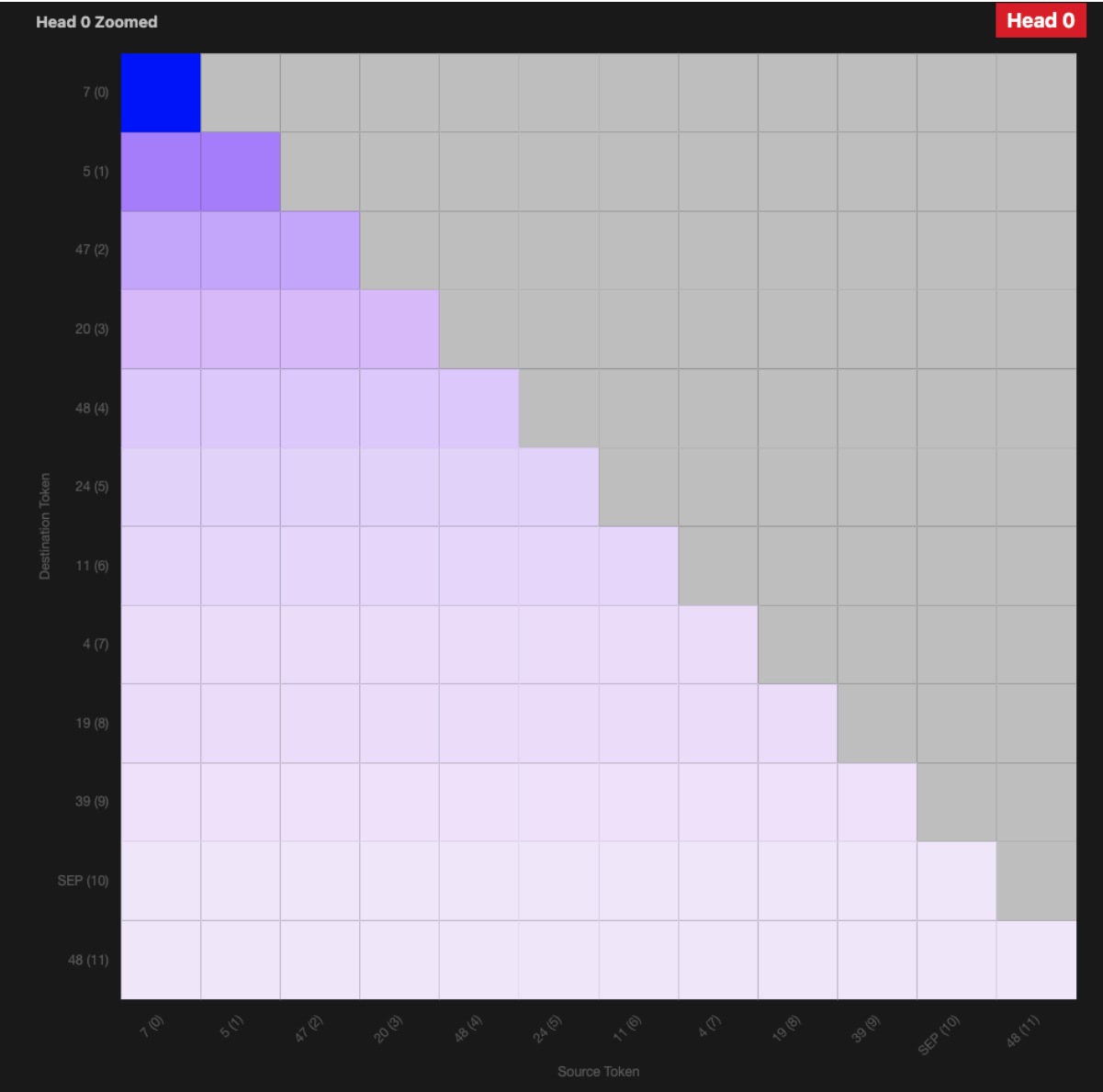
Does anyone want to take a guess

Logit Attribution

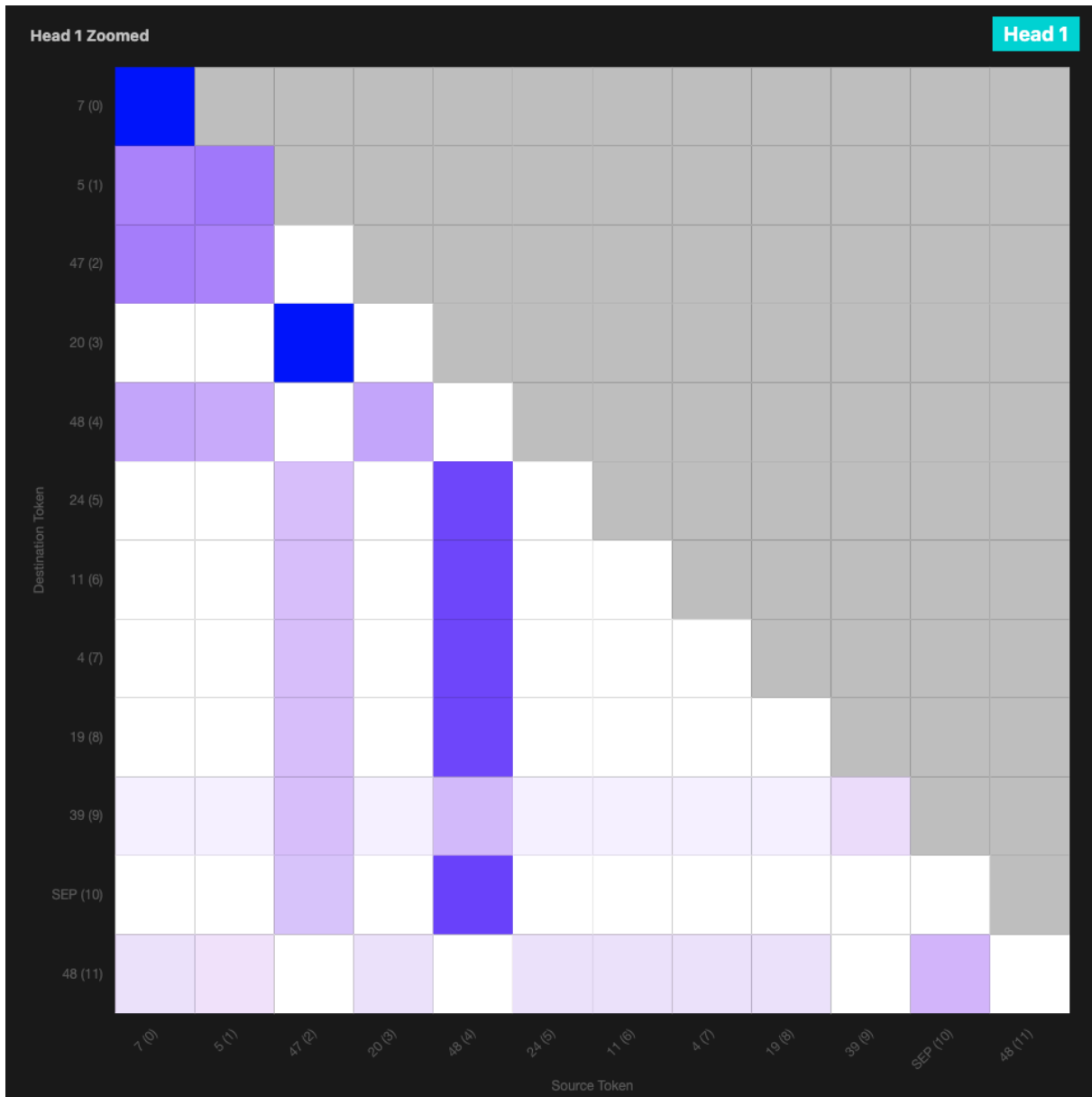




Attention Analysis

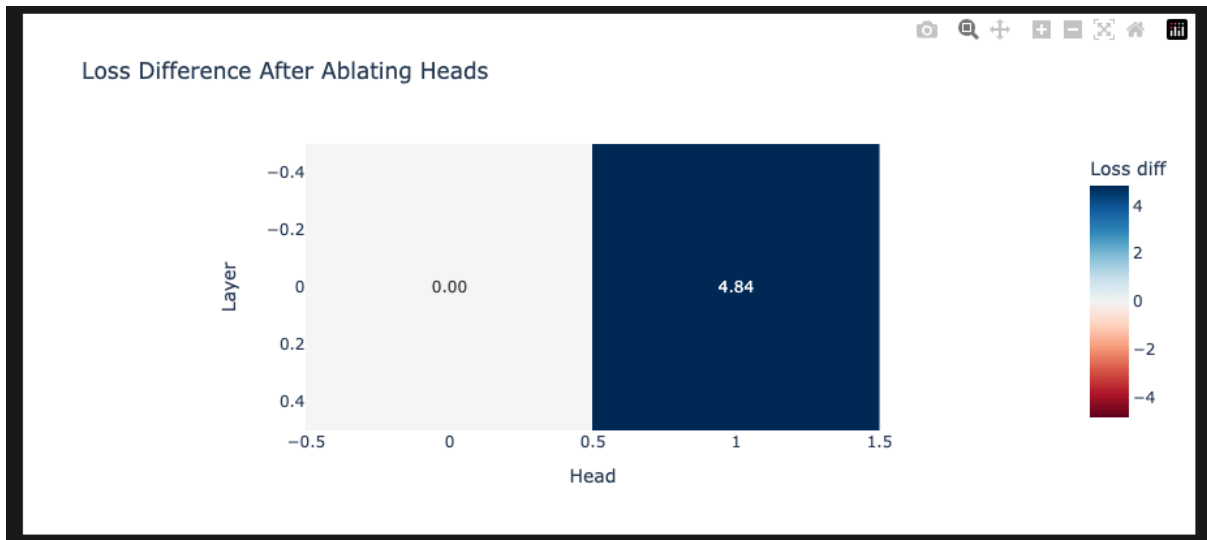


Every destination token attends to all source token



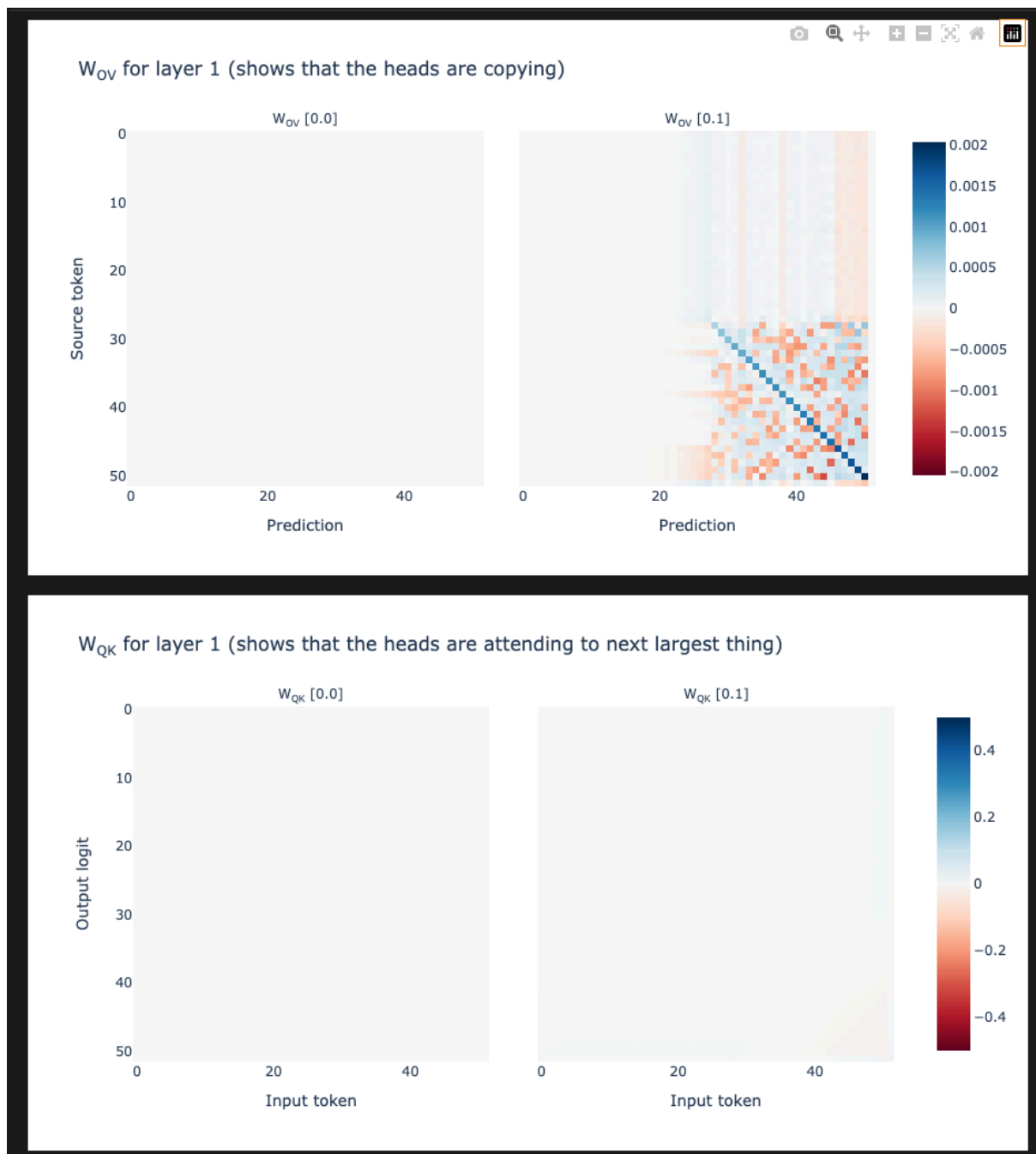
It seems every destination token just attends to the max element token, except when the element is the max token.

Zero Head Ablation



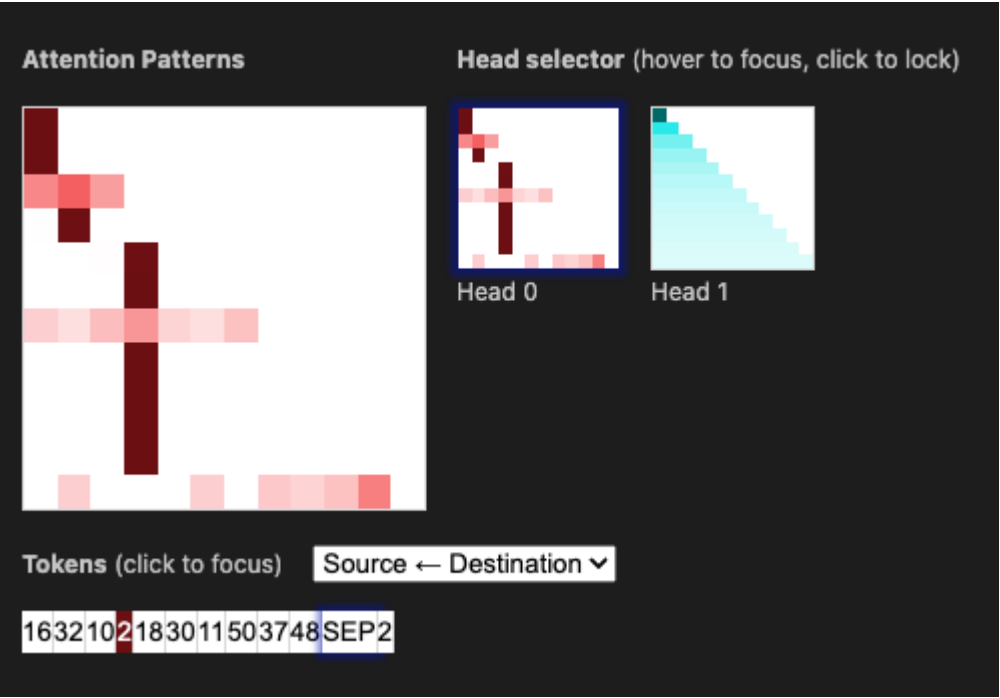
Most important head is head 0.1. So the model does this task using 1 head.

QK & OV CIRCUITS

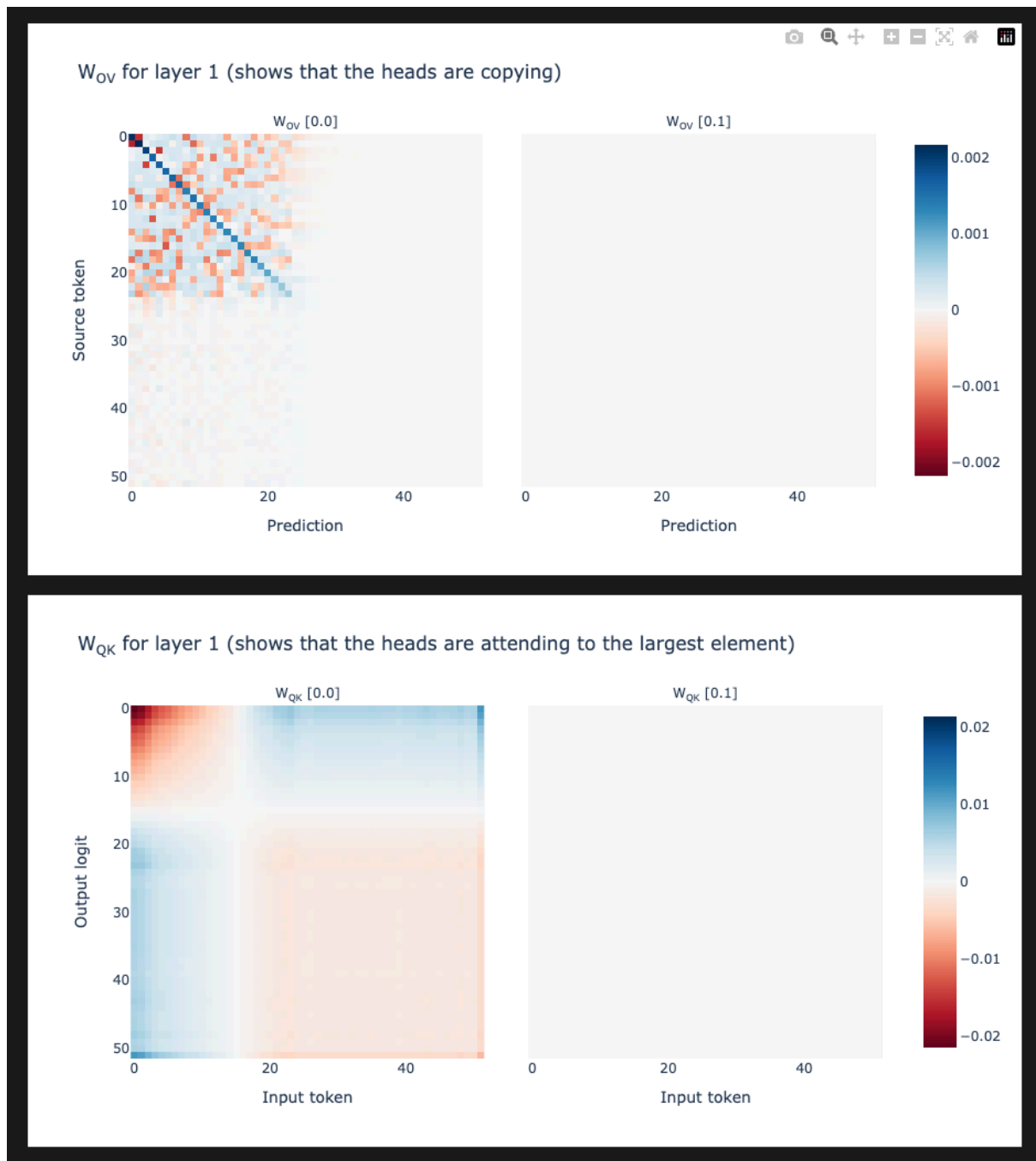


Further Exploration

Is the circuit the same for the minimum?



Model uses Head 0, I don't know if that means anything.



Future Directions

Why does the max element not attend to itself

There are also instances where the dest token doesn't attend to the max_prev_token, I'll like to know why

Why Head 1 for Max values and Head 0 for Min values

If we ablate pos_embed would performance increase?

Conclusion