Modelling Trajectories - Interim results

Introduction

Note: These are results which have been in drafts for a year, see discussion about how we have moved on to thinking about these things.

Our team at AI Safety Camp has been working on a project to model the trajectories of language model outputs. We're interested in predicting not just the next token, but the broader path an LLM's generation might take. This post summarizes our key experiments and findings so far.

How accessibly does latent space represent longer-scale concepts? How can we compress it?

TL;DR: *We try some simple probing experiments to find "text type", and it seems alright, but I think that this is likely not the best way forward for what we are trying to achieve.*

In these experiments, we try simple probing for "type of text" as a possible way to

Experiment 1: Probing Mean Latent Representations for "Genre"

We trained probes to classify the "genre" of text based on the mean activation of text chunks.

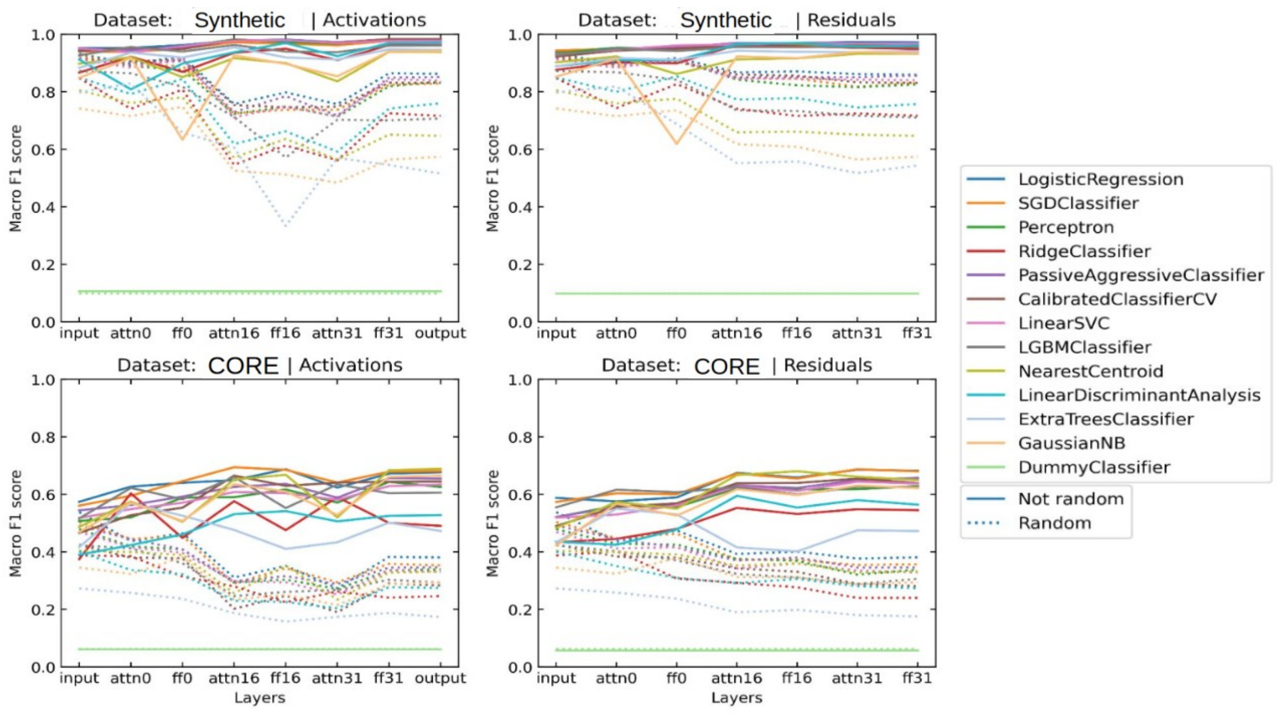
Preliminaries:

- We look at two models:
    - Mistral 7B instruct
    - We also look at randomly initialised versions of these models.
- We look at two datasets:
    - **Synthetic**: Generated outputs using Mistral using prompts like "Write a fable for children" or "Explain how to implement quicksort". We took these, passed them into ChatGPT-3.5-Turbo to be automatically labelled into classes (narrative, list, speech, code, explanation), and then these labels were manually reviewed.
    - **CORPUS:** Used the Corpus of Online Registers of English (CORE), which we modified to have fewer categories and single labels per text

Methodology:

1. Split text into chunks (initially using GPT-4, later with an algorithmic approach)
2. Computed mean activation for each chunk
3. Trained various classifier probes on these mean representations
4. Compared performance of trained vs. randomly initialized models

Results:



- Genre can be extracted with F1-scores of up to 98% and 71% depending of the dataset.
- Surprisingly small gap between trained and random model performance for probes.
- This further suggests our probes may be relying on spurious correlations rather than meaningful semantic features
- Randomly initialized models: Much worse performance, but still above chance

It's important to note that the performance gap between trained and random models was smaller than we initially hoped, suggesting our probes might be picking up on spurious correlations.

These results, while interesting, weren't as conclusive as we'd hoped. Our probes seem to be picking up on dataset-specific artifacts rather than general semantic concepts.

Can we see different 'sections' of texts appearing distinctly through attention patterns? Can we automate splitting this?

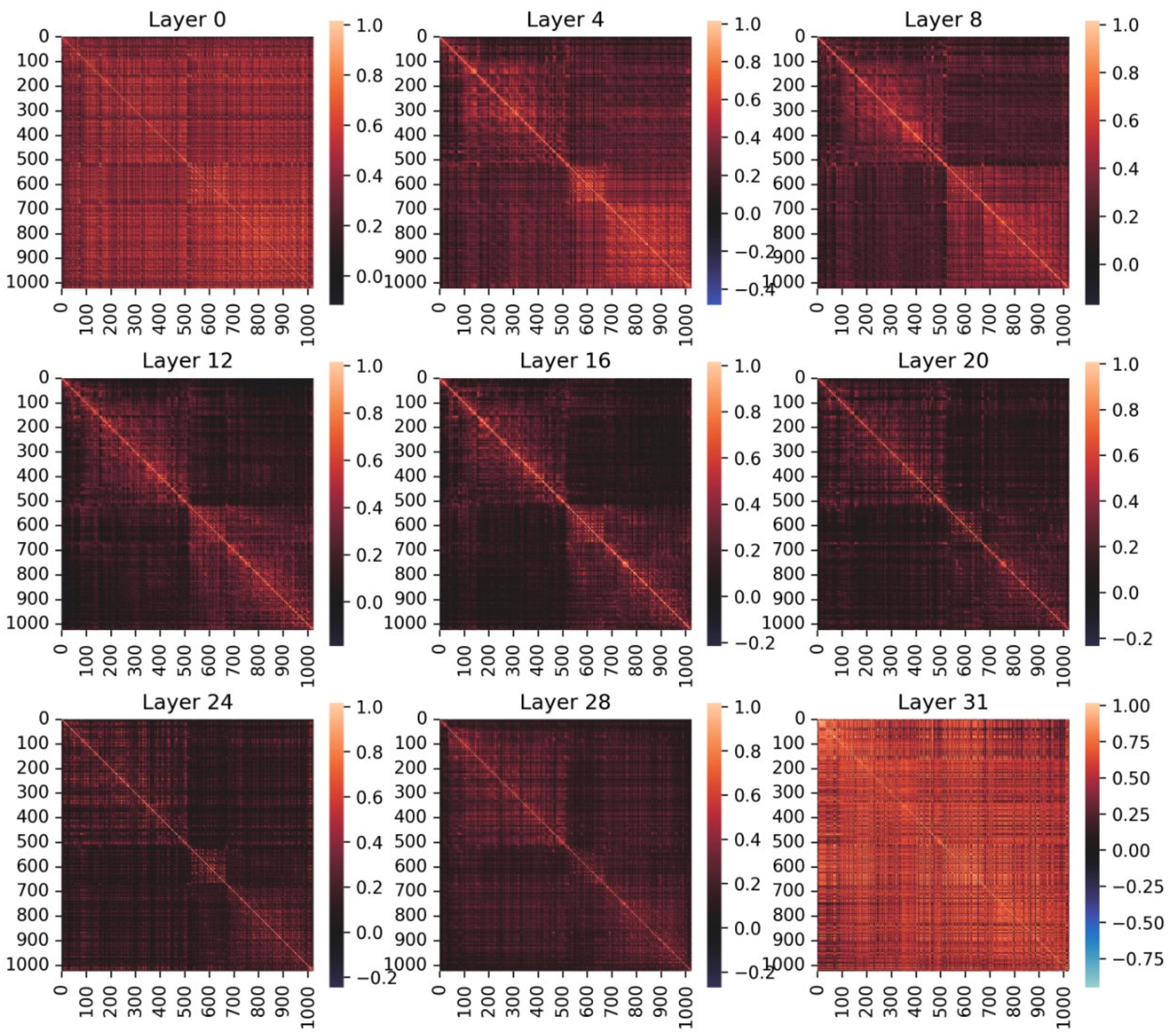
Experiment 2: Analyzing Attention Patterns

We examined attention patterns within and between different paragraphs to understand how information flows in the model.

Methodology:

1. Generated multi-paragraph texts (e.g., recipes with distinct sections)
2. Extracted attention weights from different layers
3. Visualized attention patterns using heatmaps

Results:



- Observed distinct attention patterns corresponding to paragraph boundaries
- Some layers showed clear cross-paragraph attention, while others focused more locally
- Note: This analysis was limited to a small number of texts (~2) and would benefit from a larger-scale study

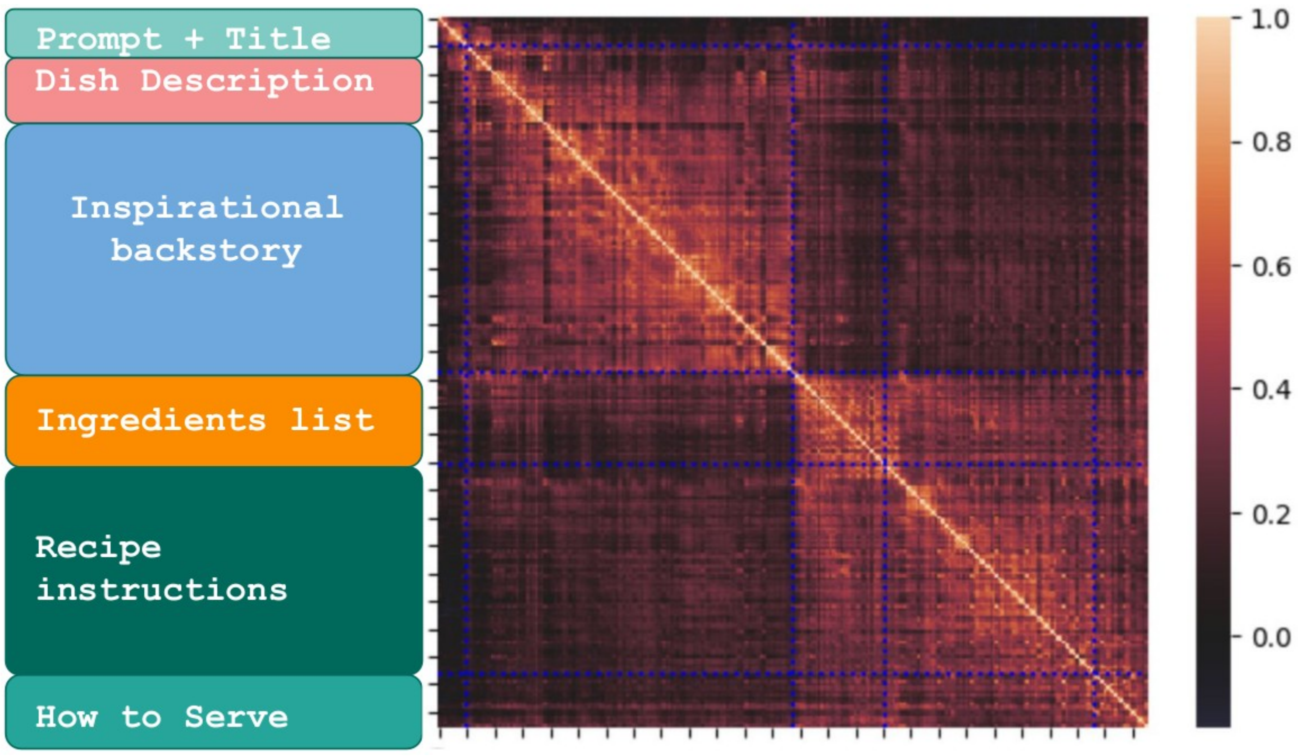
Experiment 3: Chunking Algorithm Based on Attention Patterns

We developed an algorithm to automatically chunk text based on changes in attention patterns.

Methodology:

1. Compute cosine similarity between token activations
2. Identify "breaks" where similarity drops below a threshold
3. Use these breaks to define chunk boundaries

Results:



- The algorithm successfully identified semantically meaningful chunks in an online fashion
- Performance depends on the desired level of "generality" for chunks
- While the current implementation works reasonably well, there's room for improvement in accuracy and efficiency

Can we predict future chunks of text using a simple naive method at all?

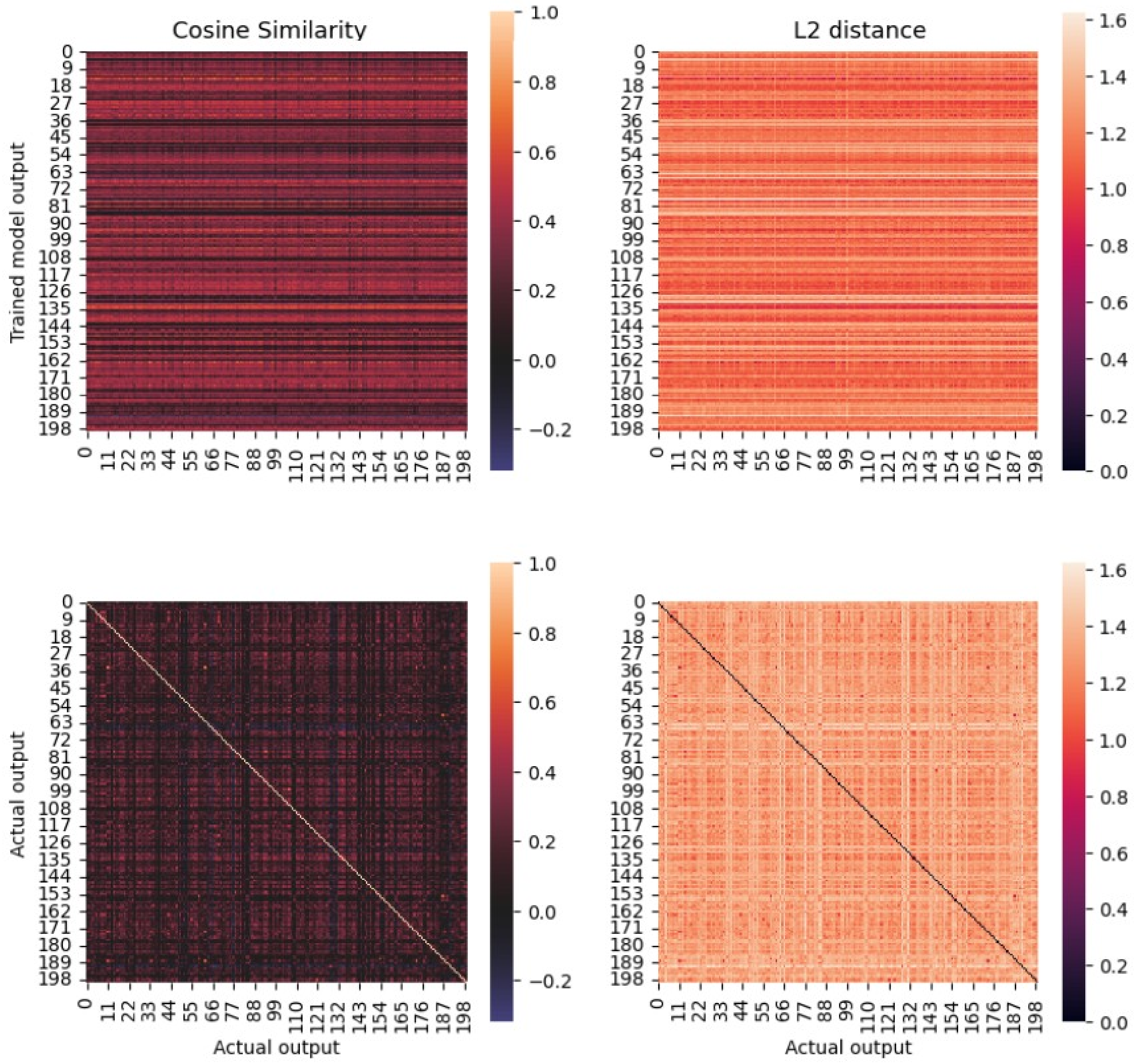
Experiment 4: Training a Predictor for Future Latent Vectors by using Activations

We used the first 100 token activations from a text to infer what would follow. Specifically, we attempted to predict either a single future token activation (the 120th token) or a set of future activations by agregating the activations from tokens 100 to 120. The activations were extracted from the Mistral model.

Methodology:

1. Created a dataset:
   - Took 100 tokens from "The Pile"
   - Generated 20 additional tokens using Mistral
2. We trained various models: simple neural networks, transformers, LLMs fine-tuned with LoRA. Either on the newly created dataset or on the full CORPUS dataset
3. We tried different methods to "aggregate/compress" the 100 input tokens in the model: calculating the average, the maximum, the sum, using PCA or using a linear layer. These methods are used at the beginning or end of the model.
4. Evaluated using cosine similarity and MSE between predicted and actual mean vectors

Results:



The top row compares model outputs with actual target outputs using cosine similarity and L2 distance. The bottom row compares actual target outputs with one another using the same metrics. The more similar the heatmaps in the top row are to those in the bottom row, the closer the model outputs are to the actual results.

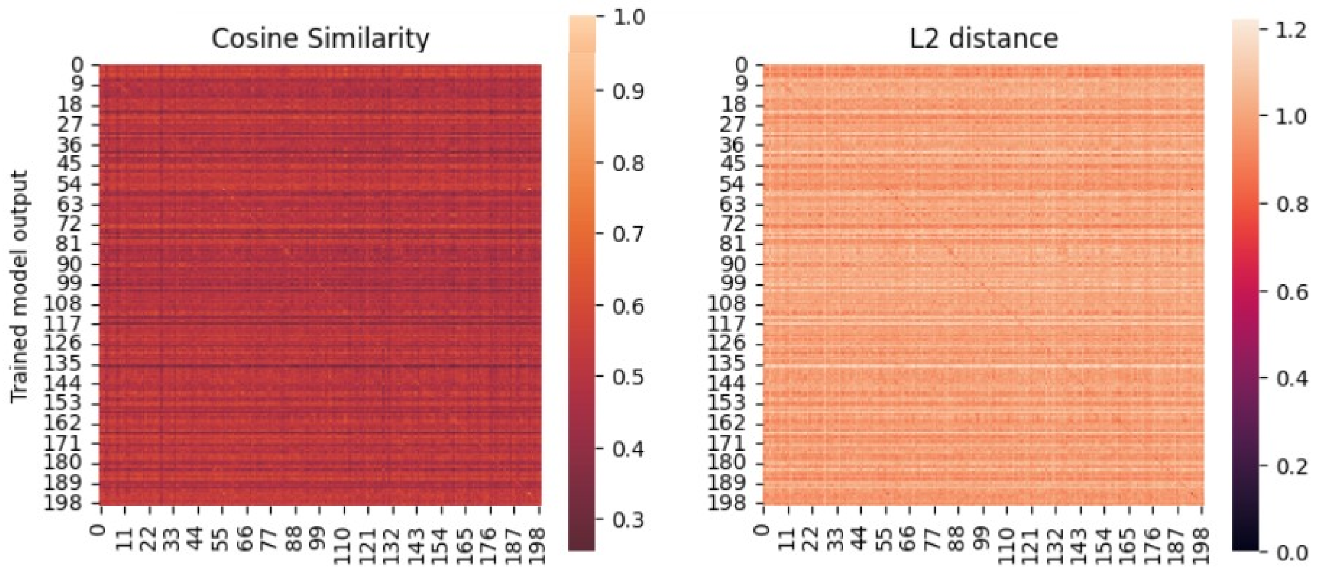
- Performance was disappointing across various configurations
- The outputs heatmaps consistently showed more prominent horizontal lines than vertical lines, indicating that the model outputs varied little with different inputs

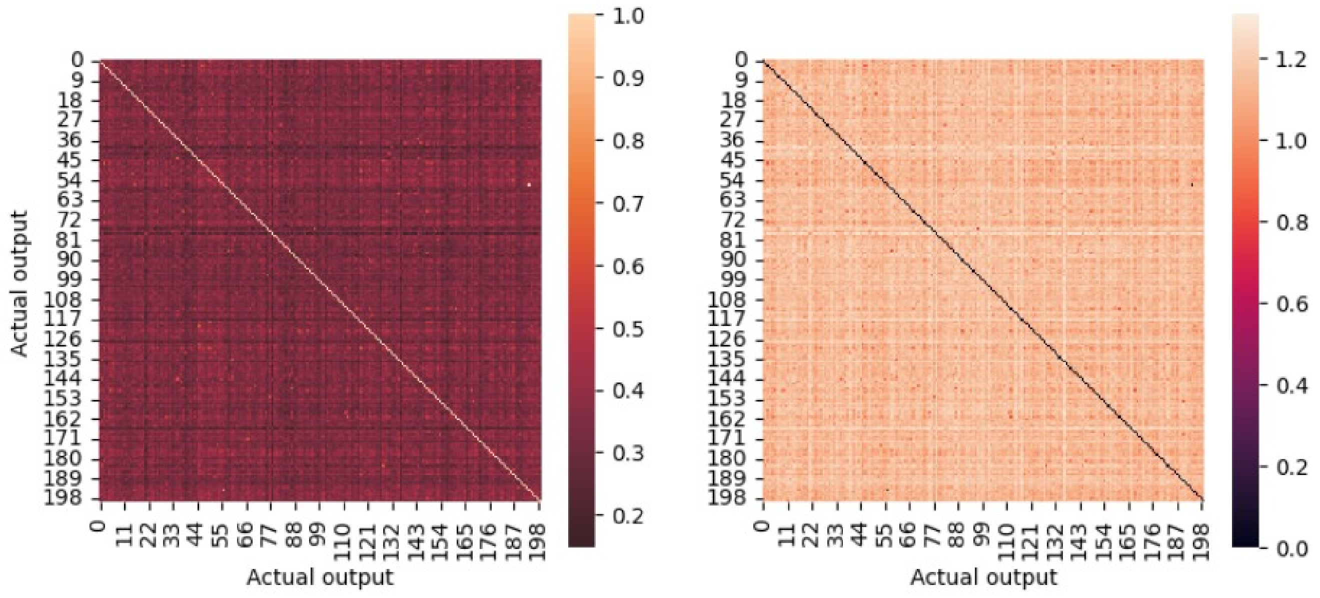
These results highlight the challenge of predicting future latent states, even with relatively sophisticated methods. Further research is needed to develop more effective prediction strategies.

Experiment 5: Training a Predictor for Future Latent Vectors by using Sentence Embedding.

Similar to the previous experiment, but instead of extracting activations from tokens, they are placed in the jasper_en_vision_language_v1 sentence embedding model.

Results:





- Strangely, when the model was single-layer neural network, the heatmaps looked better than that of a more complex model, even when the loss is lower. A single-layer model will display the diagonal and vertical lines, but if we add layers, only the horizontal lines remain
- As the precedent experiment, the outputs heatmaps consistently showed more prominent horizontal lines than vertical lines, indicating that the model outputs varied little with different inputs

Although models with lower loss values should, in theory, perform better, this improvement was not reflected in the heatmaps. This suggests that heatmaps may not be a reliable evaluation method in this context.

Conclusion and Reflections

We can envision that predicting "paragraph-scale" objects for language models has two components:

1. How do we represent these "paragraph-scale" things at all?
2. How can we map from some present token activations to some future paragraphs?

I think for our work so far, we possibly erred too much on (1) and not (2), but I am not sure. Additionally, I think we may have erred too much on running more experiments quickly rather than spending more time to think about what experiments to run. Overall, I think we have learned some valuable lessons, and I am now working on continuing these with a cohort of SPAR mentees.