Attention Phrenology: A spatial classification of attention heads

A Preprint

Giles Edkins
Alignment Jam #4
edkins@gmail.com

Keira Wiechecki
Alignment Jam #4
Department of Biology
New York University
kaw504@nyu.edu

January 22, 2023

Abstract

Even with the same architecture, input, and tokens, the internal parameters of two language models can diverge substantially. However, they can nonetheless be made to converge on the same result. We attempt to determine whether heads with similar functions can be identified in different models. We present what we believe to be a novel method of language model interpretability analysis taking inspiration from biological imaging analysis.

Keywords Mechanistic Interpretability · ML Safety · Transformers

1 Introduction

2 Methods

We trained 9 toy language models on [corpus]

We extracted weights from each head from each model in response to 1024 prompts. We used an autoencoder to compress the weights for each token pair into between 2 and 64 embedding dimensions. We used Akaike Information Criterion to select the most informative number of embedding dimensions.

2.1 Giles Methods

2.1.1 Model Architecture

9 models were trained. Each model was constructed using the **TransformerLens library. Each has the same architecture:

- A single layer
- 8 Heads
- A context window of 16 tokens
- A model dimensionality of 256
- A head dimensionality of 64
- An MLP dimensionality of 2048
- A solu activation function
- The tokenizer taken from GPT-2

2.1.2 Training

Each model was trained for 4 epochs, with a sample of 30,000 datapoints (each 16 tokens long) taken from the **Brown corpus. An **AdamW optimizer was used, and a cross-entropy loss function.

2.1.3 Attention weight extraction

A sample of 1024 datapoints (each 16 tokens long) was taken from the Brown corpus. Each datapoint was fed through each model to calculate the logits, which were discarded - instead, we captured the attention pattern of each attention head. This was saved as a big CSV file.

2.1.4 Head-distance matrix

For each head of each model (72 total), we took all the attenion values and treated them as one long vector. Then we took the Euclidean distance between pairs of these vectors, to generate a 72x72 matrix.

2.1.5 UMAP, TSNE and PCA visualizations

We ran a 2-component Principal Component Analysis, UMAP and TSNE where the datapoints corresponded to the model-heads, and the features corresponded to attention weights on the different prompts. The projected components are shown, together with projections for 3 artificial points:

- Each token only attending to itself
- Each token attending to all previous tokens equally
- Each token attending to itself and the previous token

2.1.6 PCA components

We ran a Principal Component Analysis where the datapoints corresponded to the model-heads, and the features corresponded to attention weights on the different prompts. The first 5 principal components, and the first 8 prompts, are shown.

2.1.7 How much each token attends to the first token

As a follow-up, we plotted a PCA projection of the attentions of various words in various prompts, against the first word only.

2.1.8 Head knockout

Here, further information was gathered from the models. In particular, we used the TransformerLens hook feature to erase the output from one of the heads on the given model, to investigate the effect of that head on the output. This allows, for a given prompt, producing a scatterplot of each token: the x-axis is the original logit values from the model, and the y-axis shows the logits when the head is knocked out.

The idea is that tokens above the diagonal are "suppressed" by the given head, and tokens below the diagonal are "promoted" by that head.

As a further investigation, we produced similar plots for each of the model-heads in one of the clusters, to see if they promote/suppress similar tokens.

2.1.9 Head knockout with multiple prompts

Finally, we took a selection of 64 prompts from the corpus and used the difference in probabilities (not logits) as a vector. Then we applied PCA dimensionality reduction to this and colored them according to the original clusters. The idea was to see whether clustering in the attention weights corresponds to clustering in the promoted/suppressed token space.

bottleneck	nlayer	err	aic
encode16	16	8	0.001563
44.92			
encode2	2	14	0.003355
15.39			
encode32	32	6	0.000819
78.21			
encode4	4	12	0.003071
19.57			
encode64	64	4	0.000230
144.75			
encode8	8	10	0.002481
27.99			

Table 1: AIC for encodings.

2.2 Dimension Reduction & Clustering

2.2.1 Dimension Reduction

An autoencoder is a method of dimension reduction that uses a neural network to find a lower dimensional encoding which can be decoded to recover the input. This reduces exaggeration of distance due to the number of parameters measured. We hypothesized that weights containing the most information would disproportionately contribute to the embedding.

We generated embeddings of 64, 32, 16, 8, 4, and 2 dimensions and selected the most informative using Akaike Information Criterion, which is given by

$$AIC = 2k - 2ln(L)$$

where k is number of parameters (embedding layers in this case) and L is a loss function (MSE in this case).

2.2.2 Clustering

A 9-nearest neighbors graph is constructed from euclidean distance calculated from the encoding layer.

The leiden algorithm attempts to find a clustering that maximizes modularity H for a given graph and resolution γ. Modularity is defined as

$$H = \frac{1}{2m} \sum_{c} (e_c - \gamma \frac{K_c^2}{2m})$$

where m is the average degree of the graph, e_c is the number of edges in cluster c, and K_c is the number of nodes in cluster c. This gives a measure of how well-connected clusters compared to expectation based on average degree of the graph and number of nodes in a cluster. A higher γ results in more clusters.

We obtained 1000 clusterings on random γ values between 0.01 and 3. We assessed clustering by several metrics. To identify heads performing similar functions, we attempted optimise for there to be exactly one head from each model in each cluster. To select the resolution parameter, we calculated an F-score using the first head from each model in a cluster as true positives and each subsequent head from the same model as false positives.

3 Results

3.1 Giles Results

In 7 we see, obviously, that each head is identical to itself (the diagonal). We see some heads that stand out as being generally different, but similar to each other (the horizontal and vertical yellow lines). These roughly correspond to heads that mostly have each token attending to itself. Some other heads are a little closer or further than average but none are identical in their behavior.

In 8 we see the black dot (corresponding to attending to everything equally) is central to one of the clusters of heads. The two grey dots (dark grey corresponding to attending to the current token, and light grey corresponding to attending to the current and previous tokens) are less central but still fairly close to real head clusters.

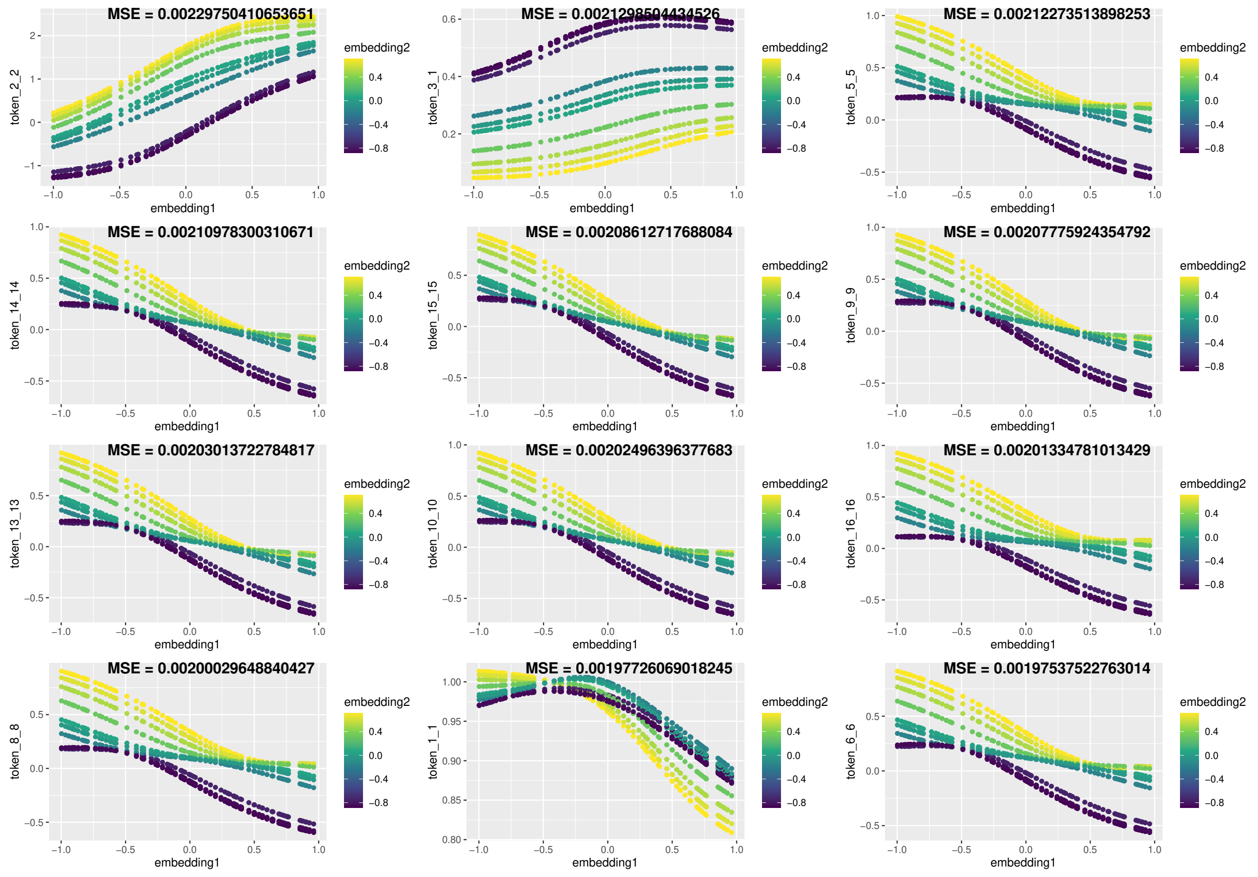

Figure 1: Decoded token pair weights generated by embedding value pairs. MSE values were calculated based on how well the autoencoder recovered the input when the indicated token pair was randomized in the input.

In 9 we see the first principal component has entries along each of the diagonals, corresponding to attending to the current token.

The second principal component corresponds (roughly) to attendig to the previous token, although there are some wiggles.

The remaining principal components exhibit different behaviour for each prompt, and warrant further investigation. The third principal component is shown in 10. This shows one sign for the first column (attending to the first token in the sequence) and the opposite sign for the remaining tokens. Furthermore, the sign is positive (red) for common "stop-words" such as "the", "of" and "and" and negative (blue) for more semantically meaningful words.

This effect is explored further in 11, where we see how much each token attends to the first token, across all model-heads, but reduced to 2 dimensions with PCA. This shows a clear cluster of capitalized words up the top-right, and stop-words towards the lower-right of the main cluster.

3.2 Knocking out heads

For the first prompt that we investigated, and for the first head of the first model, the clearest result came after 6 tokens. The prompt was "They sit alone in their rooms", and 12 shows that words such as "are", "were" and "will" are suppressed by the given head. Grammatically, the words should not be there (or are very unlikely), and the model has not fully learned this. However, the given head is in a sense "helping" the others out here.

In 13 we extend this investigation to multiple models and heads - specifically everything in cluster 2. Some of the other models' heads seem to have a similar purpose, suppressing tokens such as "are" and "were". But not all of them.

Figure 14 shows how this effect extends across multiple prompts and all clusters. We see the turquoise cluster scattered across the bottom, the green cluster over to the left, and the peach/pink/gold clusters (which were near each other in

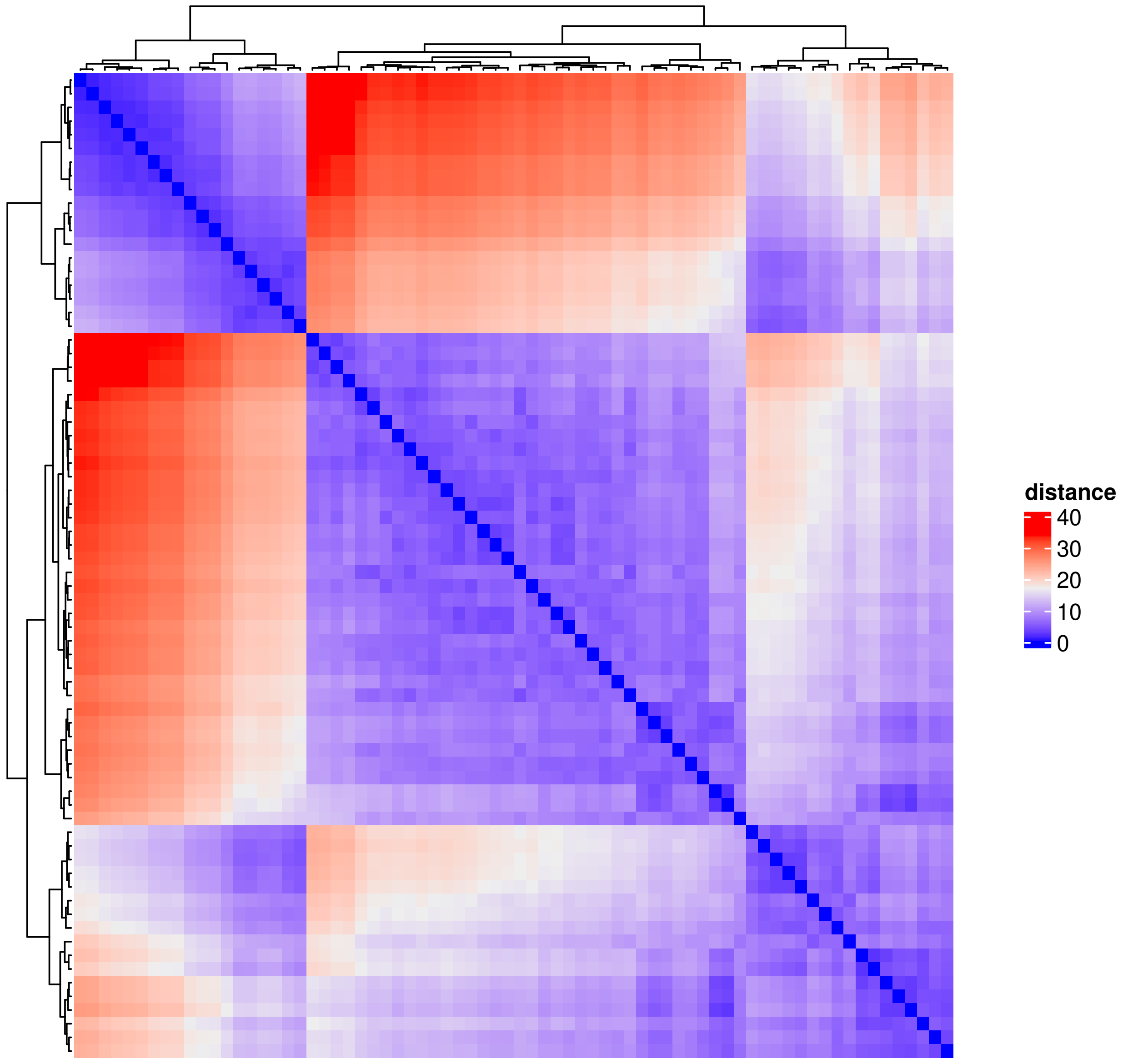

Figure 2: Pairwise euclidean distance between attention head embeddings.

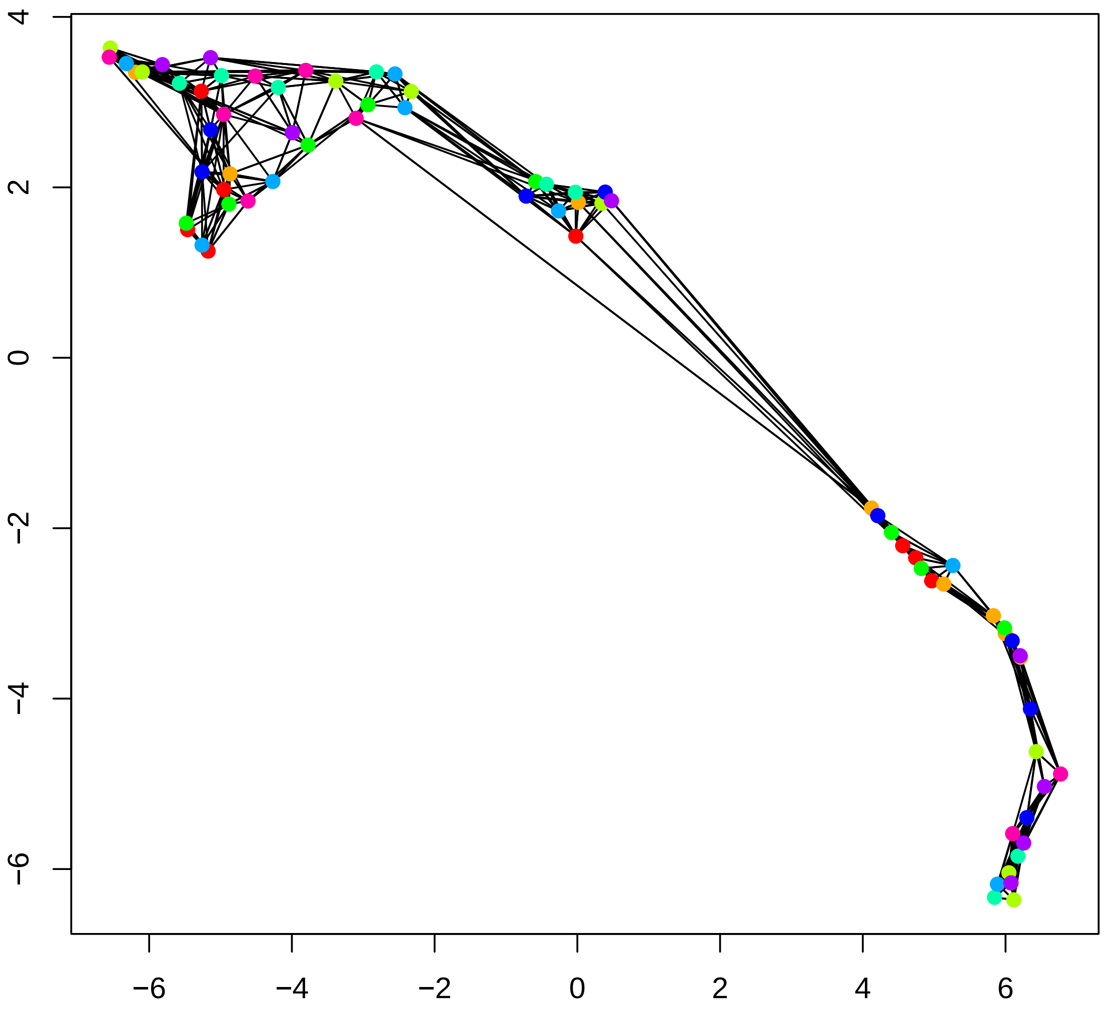

Figure 3: 9-nearest neighbors graph of attention head embeddings. Heads are color coded by model.

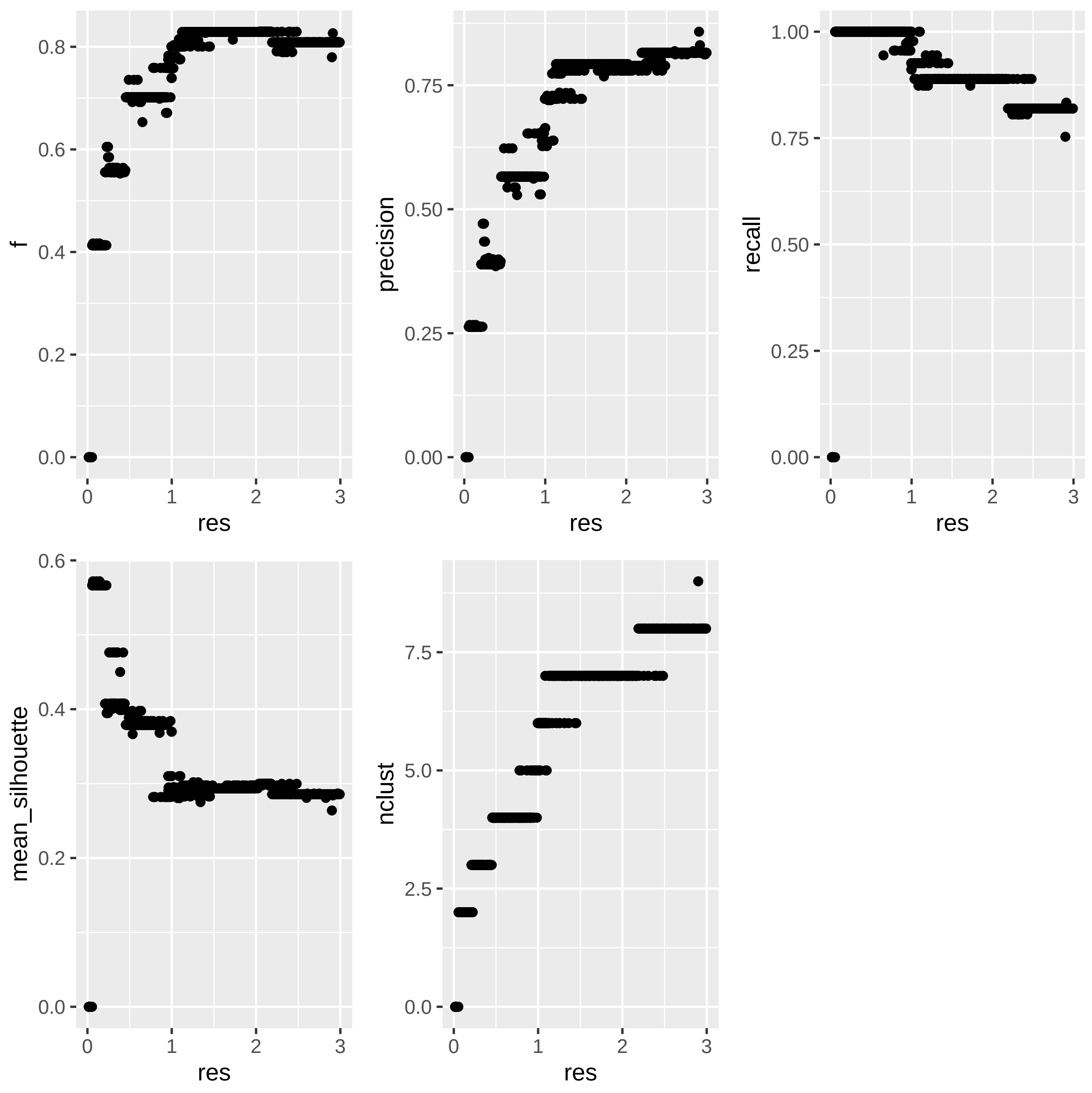

Figure 4: Optimization of γ selection.

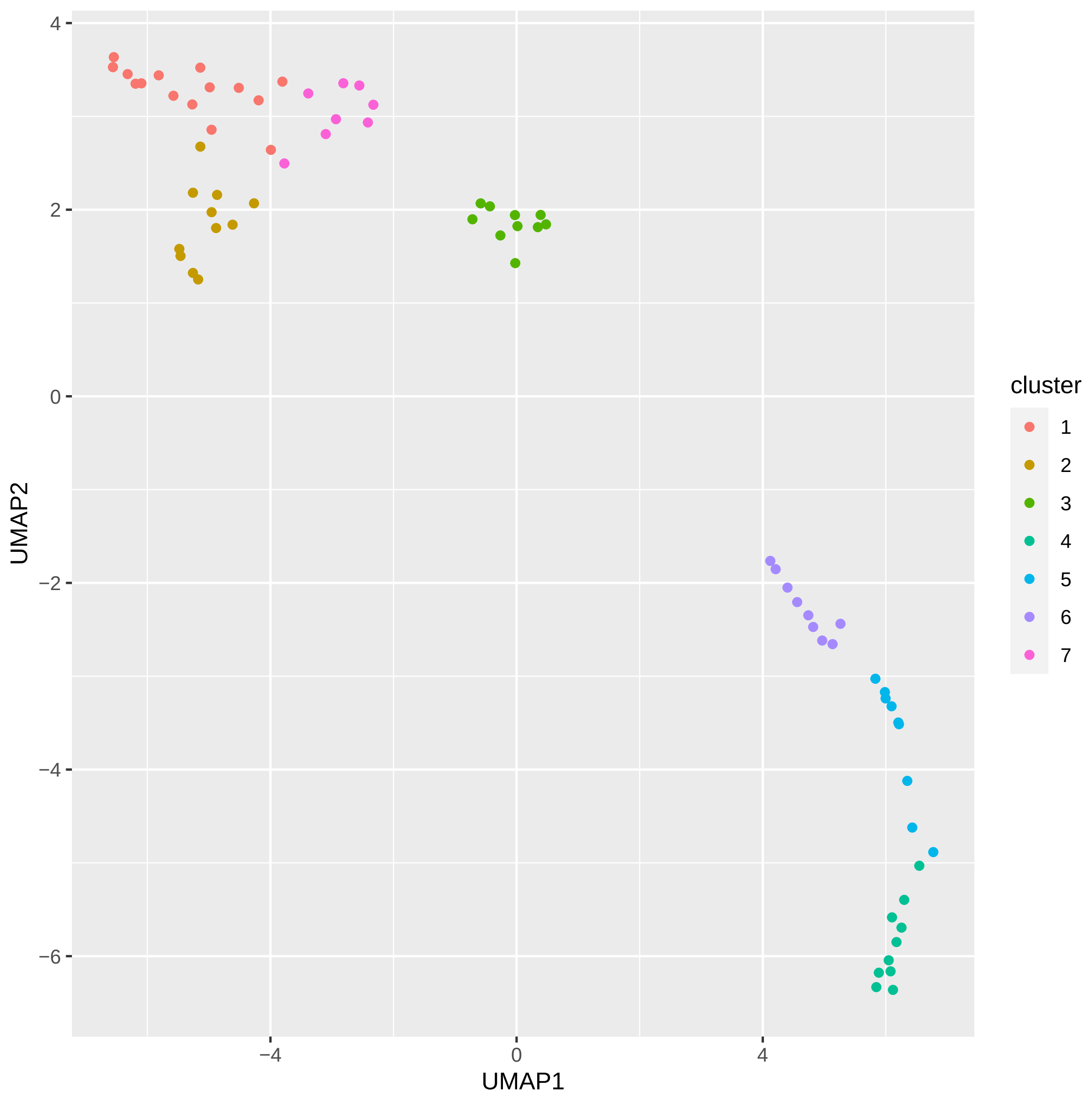

Figure 5: Optimal clustering at $\gamma = 2.16$.

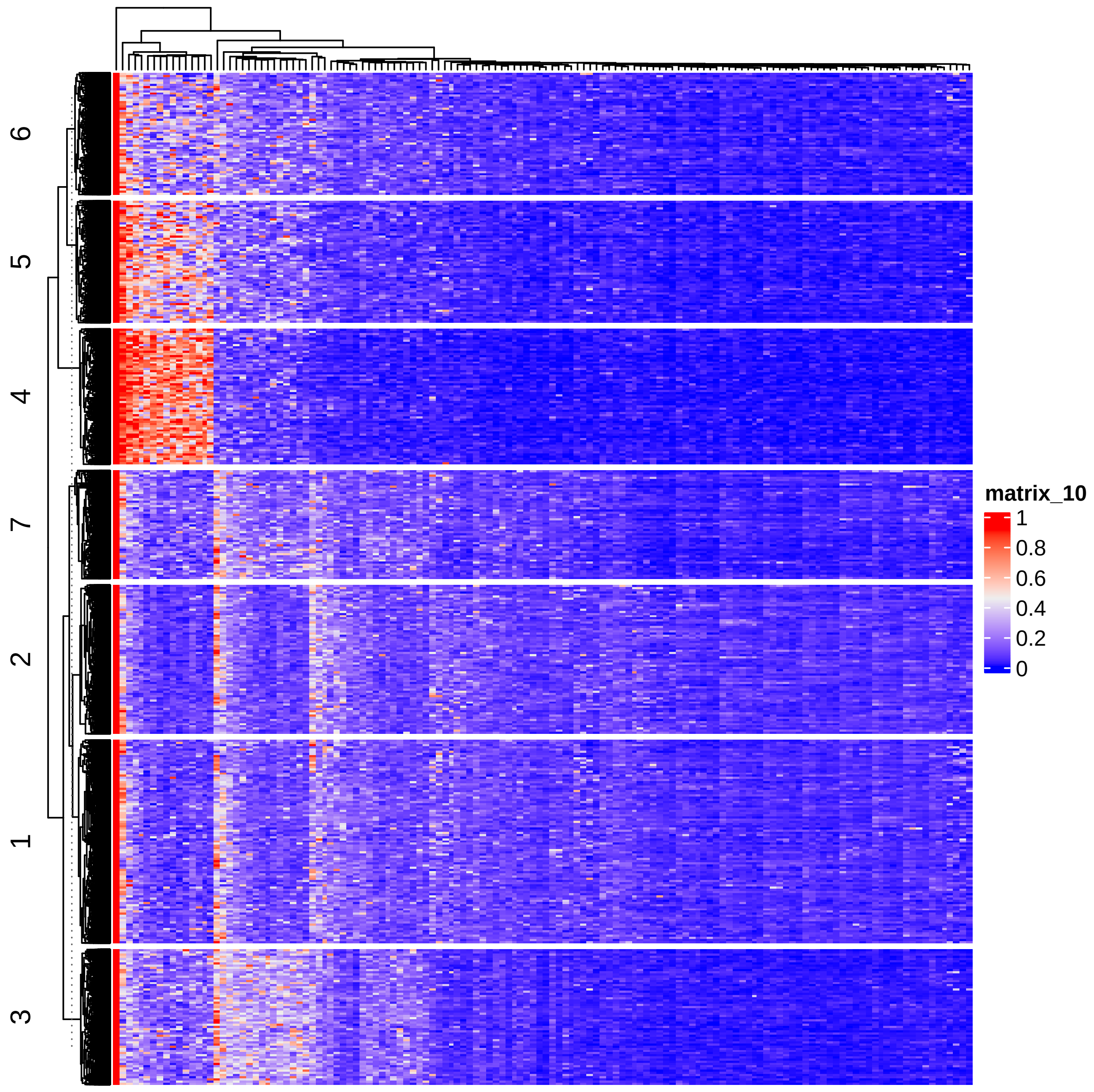

Figure 6: Token pair weights by cluster.

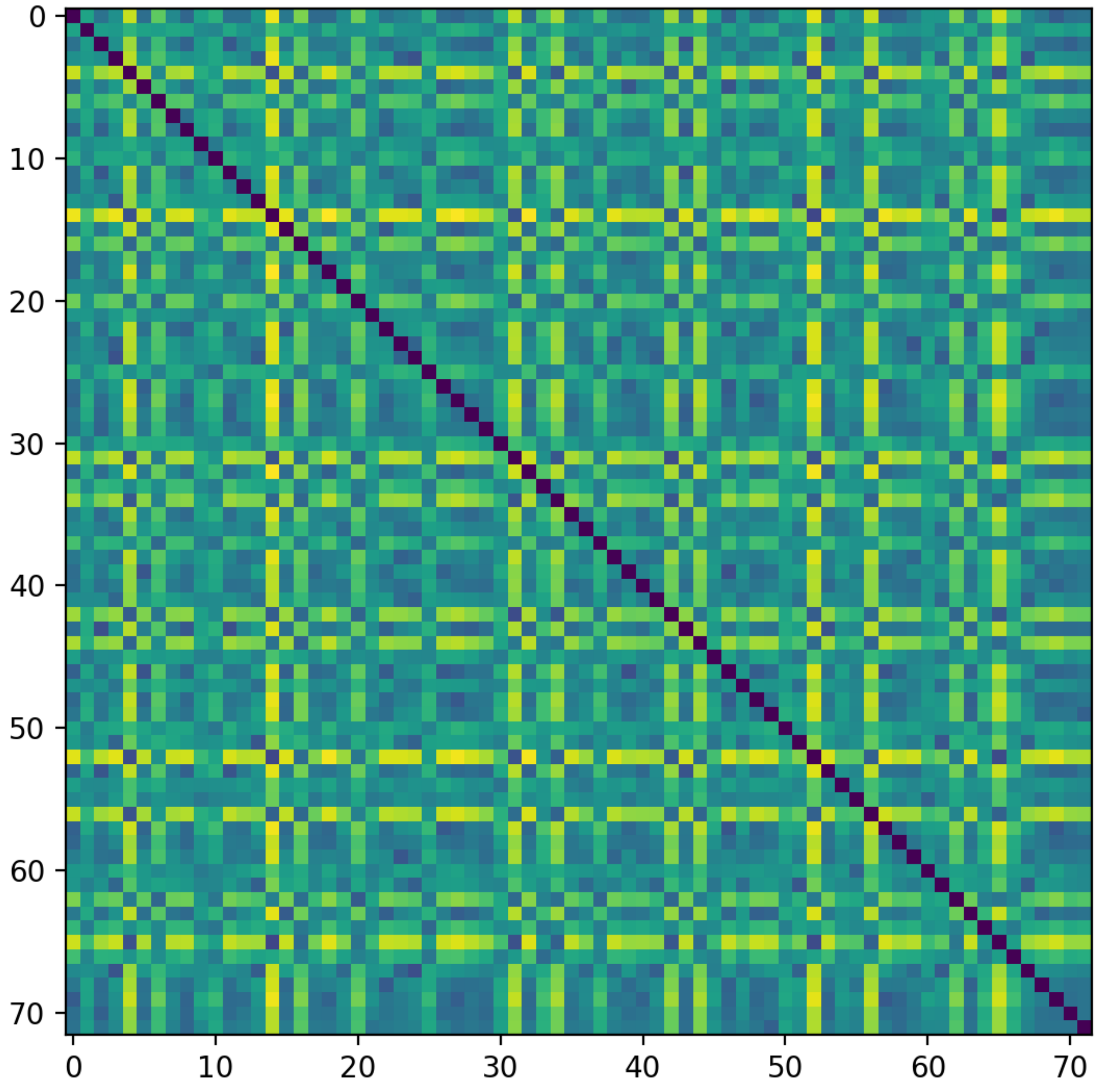

Figure 7: Pairwise euclidean distance between attention head embeddings, arranged in the original order.

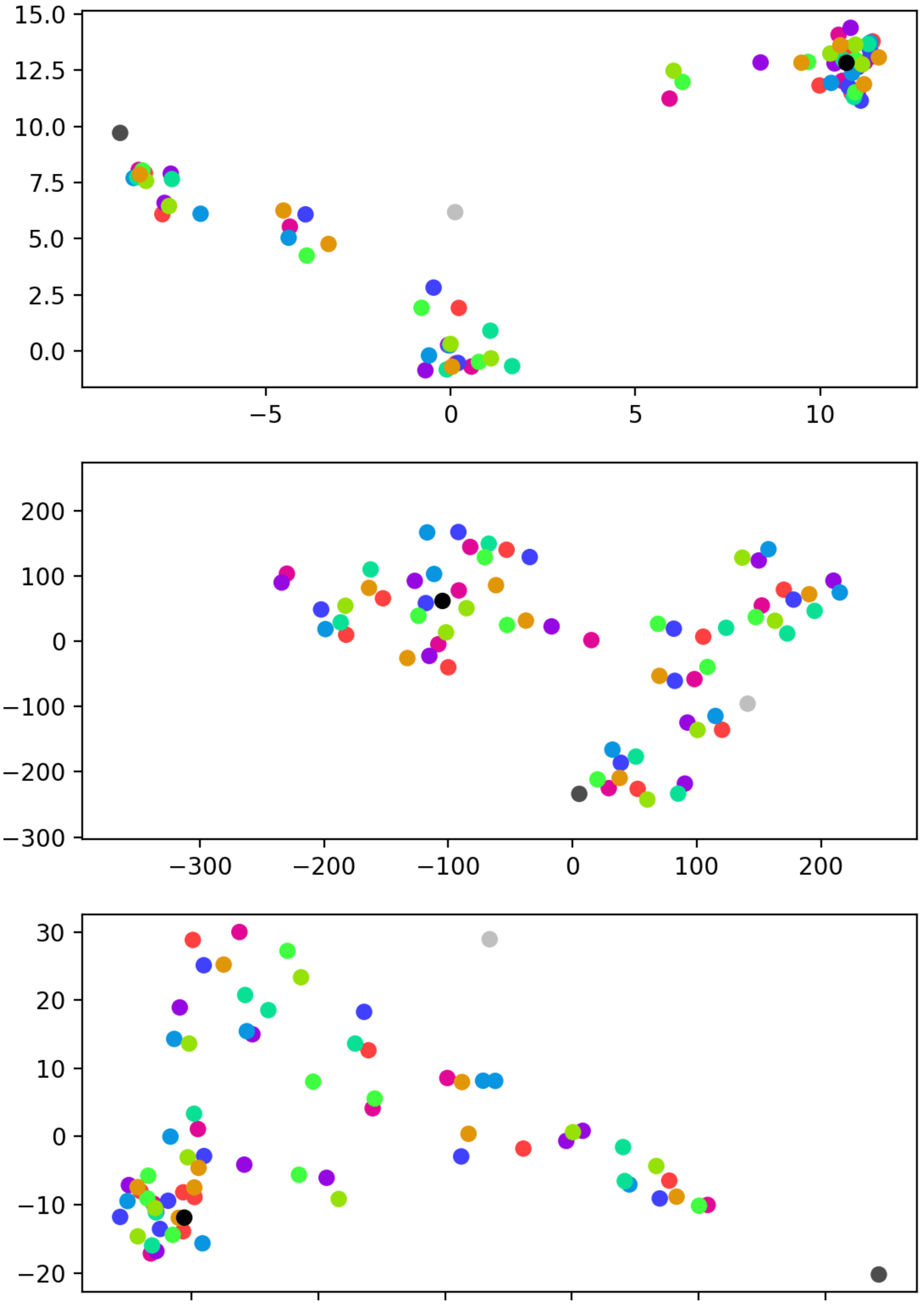

Figure 8: UMAP, TSNE, PCA dimension reductions of attention head embeddings together with artificial points in grey.

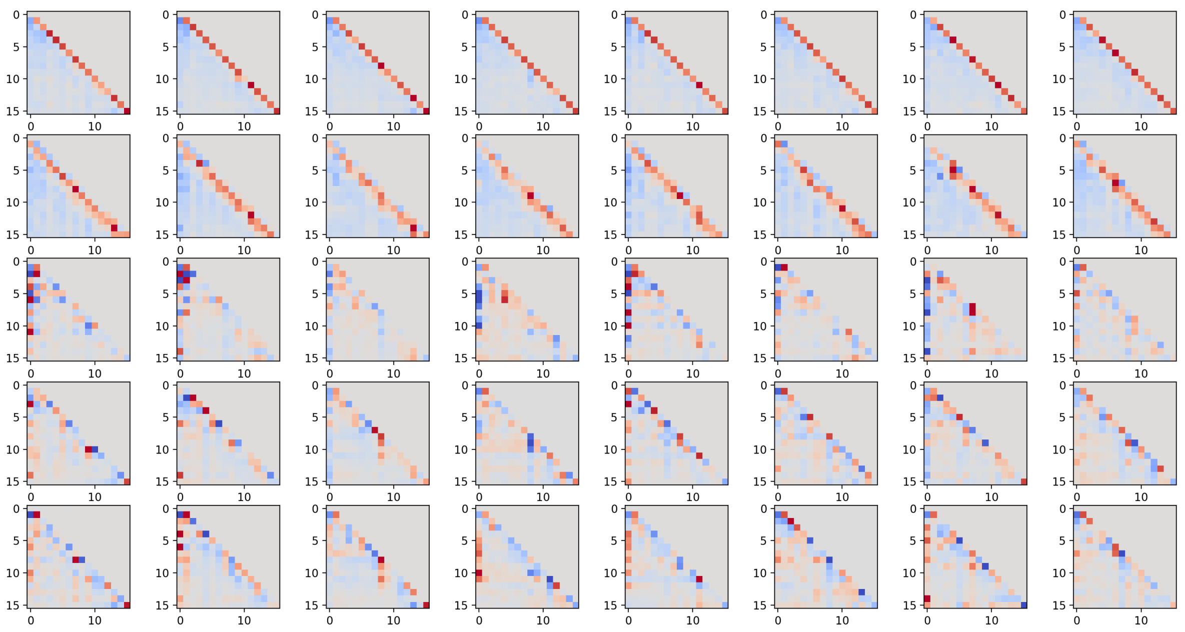
Figure 9: PCA components of attention head embedding

the original plot) mixed together nearby. This suggests that for some clusters, there is a relatedness that goes beyond just similar attention patterns - they also have similar effects on the output.

4 Discussion

By comparing separately trained toy models of with the same architecture, we were able to perform functional analysis of attention heads in isolation. We believe this to be a promising and underexplored direction in mechanistic interpretability. We were able to assess which token pairs were informative by using dimension reduction and random perturbation of input data. Our approach was limited by the small size and limited training of our models, but we believe it will scale.

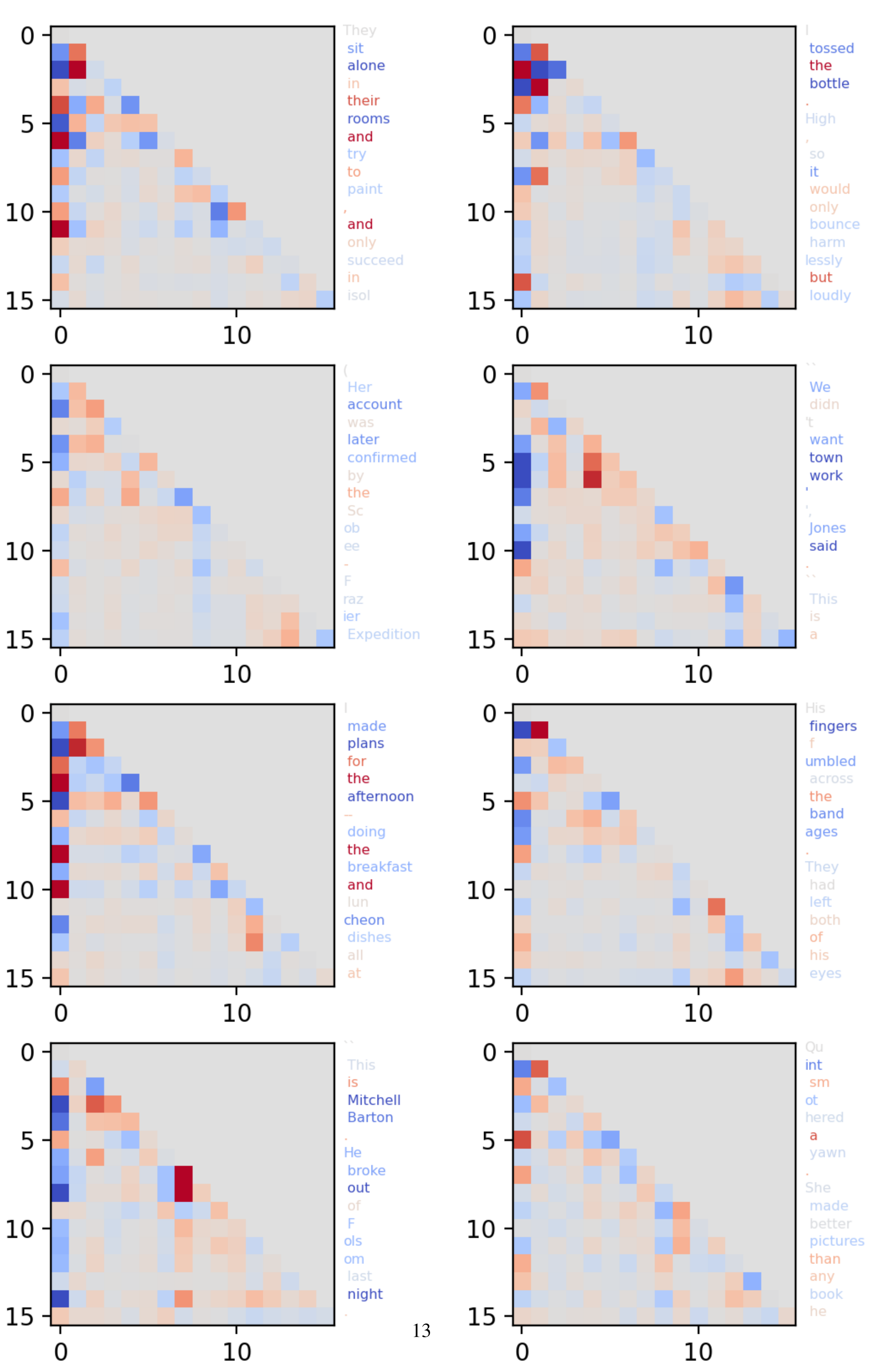

Figure 10: Third PCA component annotated with prompt tokens

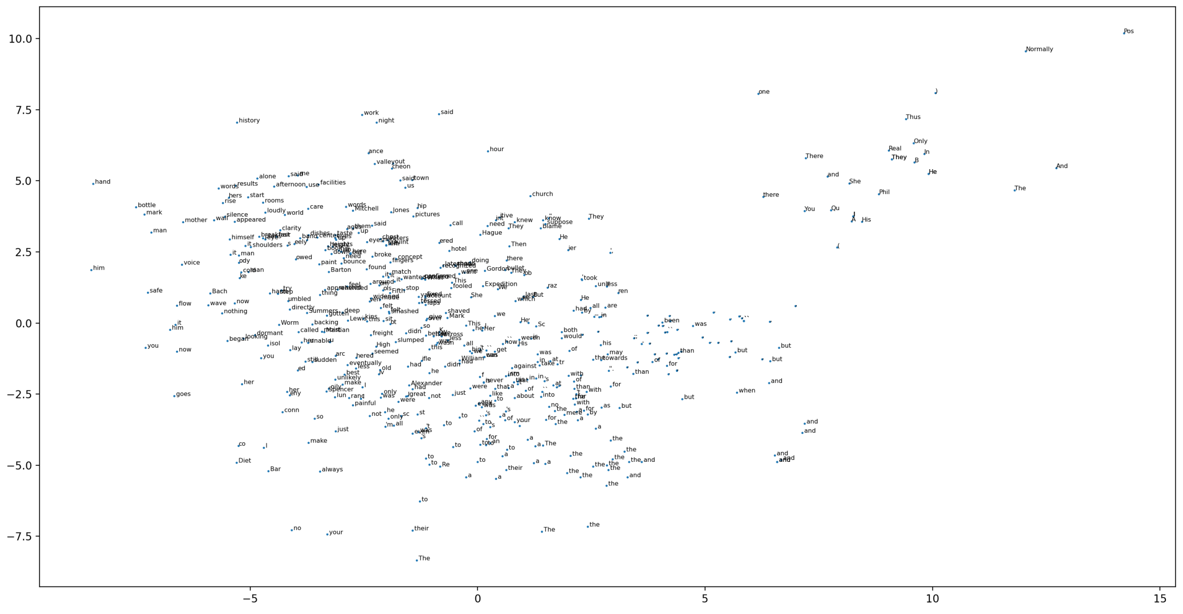

Figure 11: PCA dimension reduction of how much each token attends to the first token in the prompt

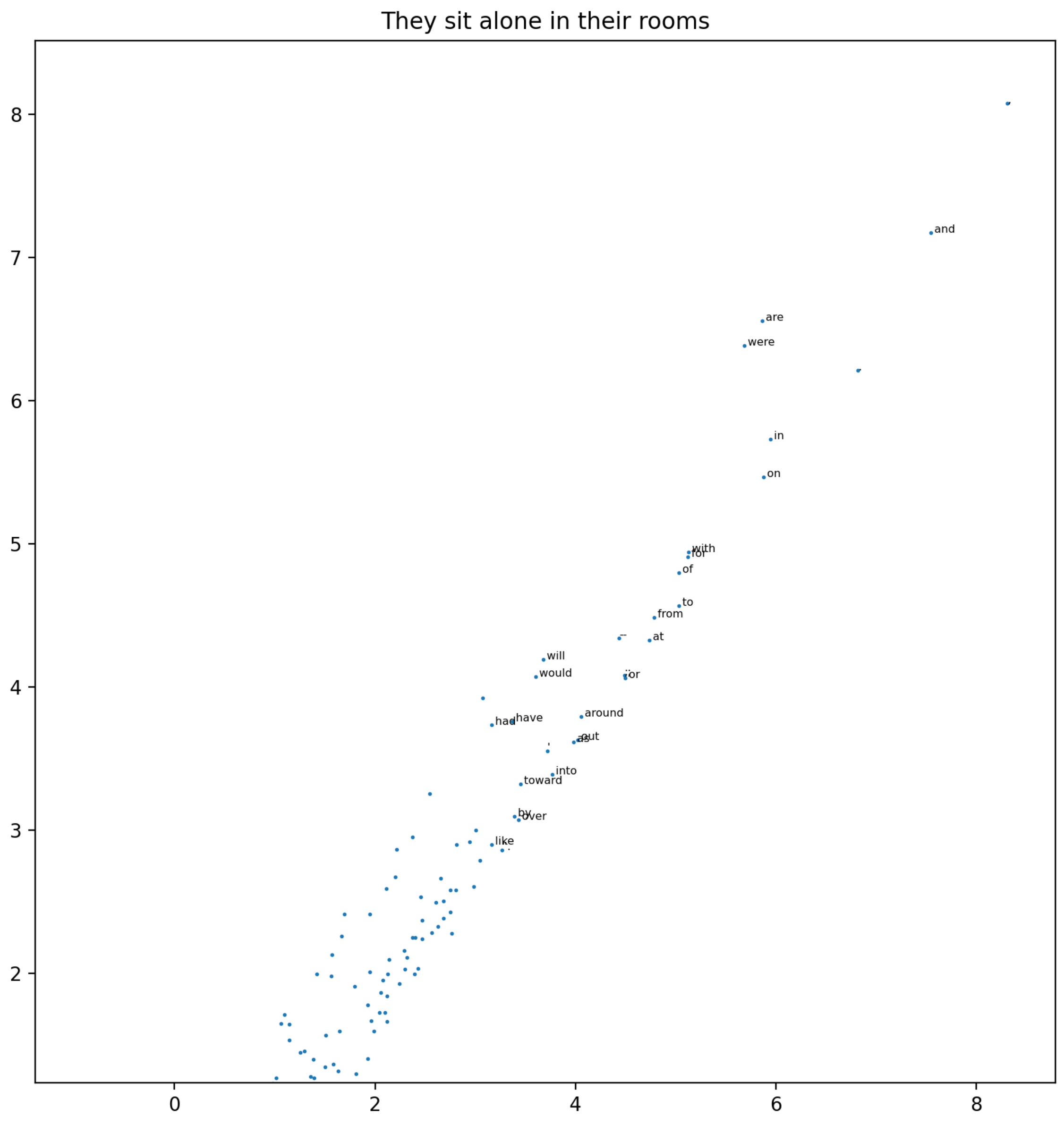

Figure 12: Scatterplot showing token logits emitted for the last token in the given prompt. X axis: original model. Y axis: model with head 1 knocked out.

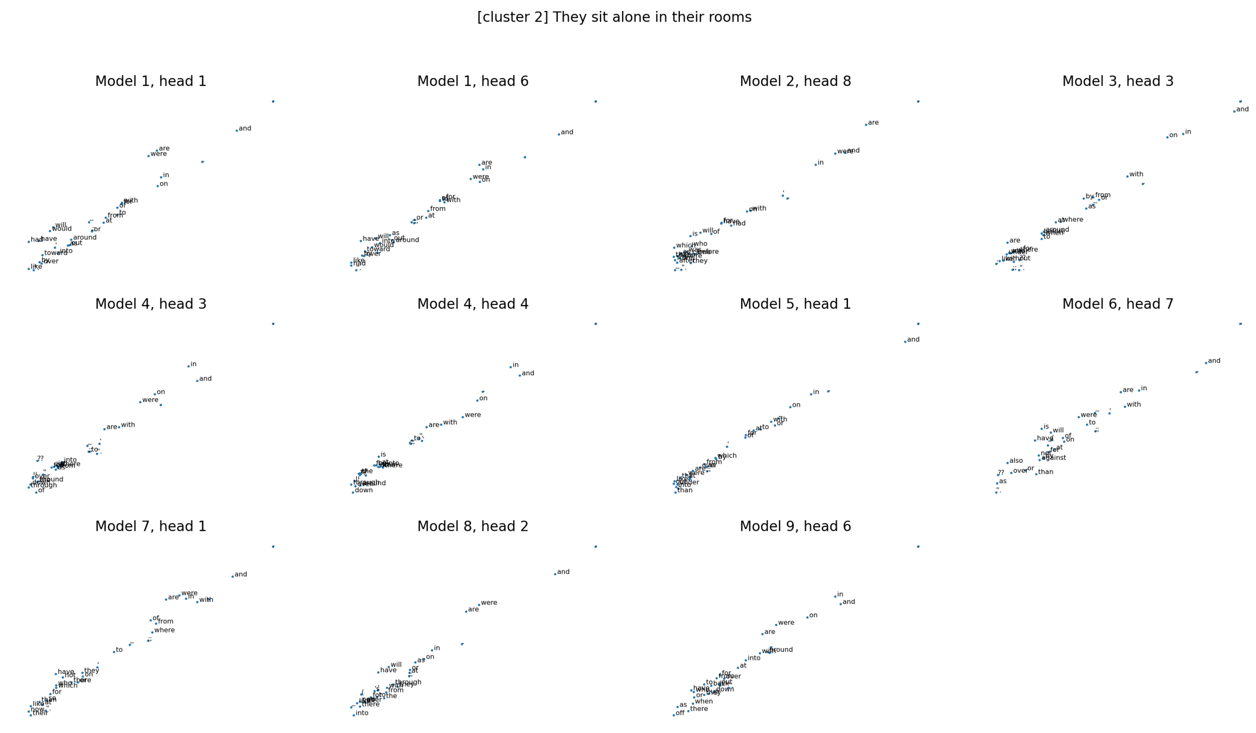

Figure 13: Scatterplot showing token logits emitted for the last token in the given prompt, for multiple models and heads in the same cluster. X axis: original model. Y axis: model with the given head knocked out.

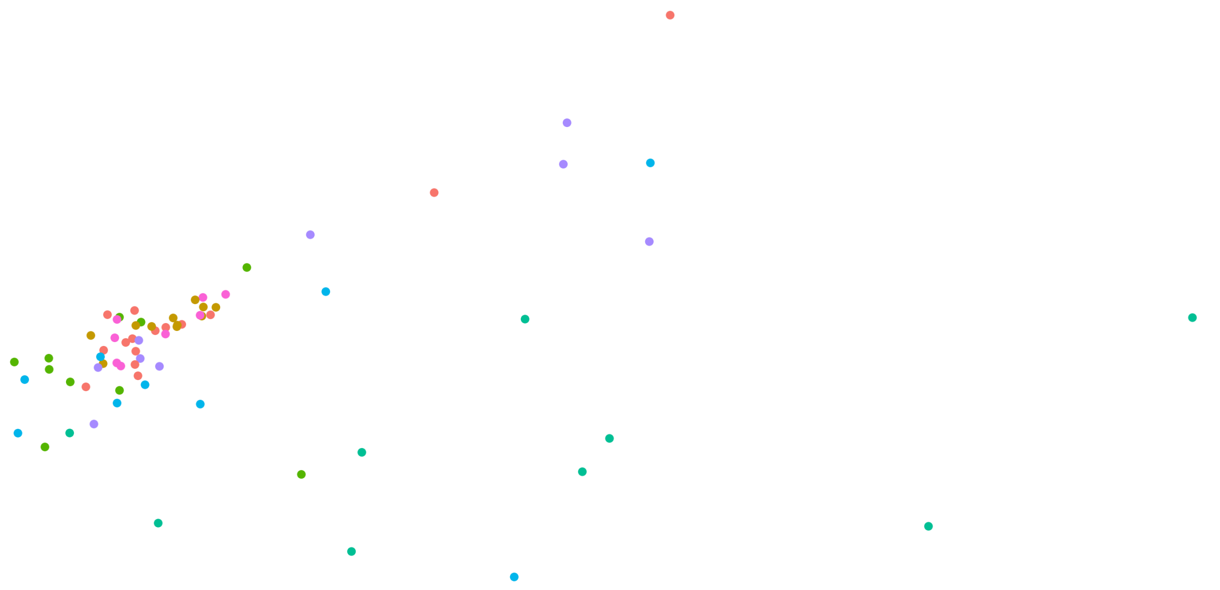

Figure 14: PCA dimesnion reduction of model-heads, colored according to original clustering and positioned according to differences in probabilities when the given head is knocked out.