Steering Model's Belief States[1]

Gonçalo Paulo
EleutherAI

Sinem Erisken
Independent

Tassilo Neubauer
Independent

Organized by
Adam Shai, Paul Riechers, PIBBSS

Abstract

Recent results in Computational Mechanics show that Transformer models trained on Hidden Markov Models (HMM) develop a belief state geometry in their residuals streams, which they use to keep track of the expected state of the HMM, to be able to predict the next token in a sequence. In this project, we explored how steering can be used to induce a new belief state and hence alter the distribution of predicted tokens. We explored a traditional difference-in-means approach, using the activations of the models to define the belief states. We also used a smaller dimensional space that encodes the theoretical belief state geometry of a given HMM and show that while both methods allow to steer models' behaviors the difference-in-means approach is more robust.

Keywords: Computational Mechanics, Steering.

1. Introduction

As advancements in AI keep on accelerating and the use of AI becomes more ubiquitous, researchers are scrambling to minimize AI risk and ensure safety and utility for humanity. As AI systems become more powerful, it is essential that we are able to guarantee that these models remain aligned with users' goals and values.

One promising research avenue involves steering Transformer's behaviors by manipulating their inner functioning. For example, it is possible to force chatbots to not refuse prompts [1] , be less sycophantic [2] or reply as if they believe that they

[1] Research conducted at the Computational Mechanics For AI Safety Hackathon, 2024 (https://www.apartresearch.com/event/compmech)

are bridges [3]. Steering is usually possible by interrupting computation in the residual stream at certain layers and making specific interventions.

Recent work applying Computational Mechanics to Transformer models trained on Hidden Markov Models (HMMs) showed that these Transformers encode belief states into their residual stream [4]. This means that not only do Transformers have a model of how the HMM generates outputs, but they also have a mechanism to update their beliefs and synchronize with the world state. Since the belief structure is geometrically embedded in the residual stream, we hypothesize that it should, in principle, be possible to not only steer the model's beliefs and control the output but also investigate how these steering methods align with respect to the observed internal geometry.

In this project, we present evidence that the geometric structure found in the residual stream of the transformer contributes to encoding the beliefs of the model by actively changing this structure and observing that the model behavior can be controlled. We do this by both creating steering vectors using the model activations and by using vectors inspired by the internal geometry that can be used to change the probability distribution of the output tokens of the model to a desired state.

2. Methods

2.1 Hidden Markov Process

In this project, we used a toy model Transformer trained to predict an RRXOR process (Figure 1a). This process only emits one of two tokens [0,1] with a probability dependent on the current (hidden) state. If we want to predict the process, rather than simply generate the emissions, we need more than just the generative HMM - we need a Mixed-State Presentation (MSP; Figure 1b) . It is in fact the MSP, and not simply the generative representation of the underlying HMM, which is present in the residual stream of Transformers [4].

The RRXOR has 36 distinct states in the MSP, and has 5 recurrent states, corresponding to the 5 generative states. In general, this means that the state believes can be encoded in a 4-simplex.

2.1.1 The Transformer model

We used a pretrained model provided by the Computational Mechanics For AI Safety Hackathon team. This Transformer has 4 layers, a hidden dimension of size 128 and was trained with a context length of 10.

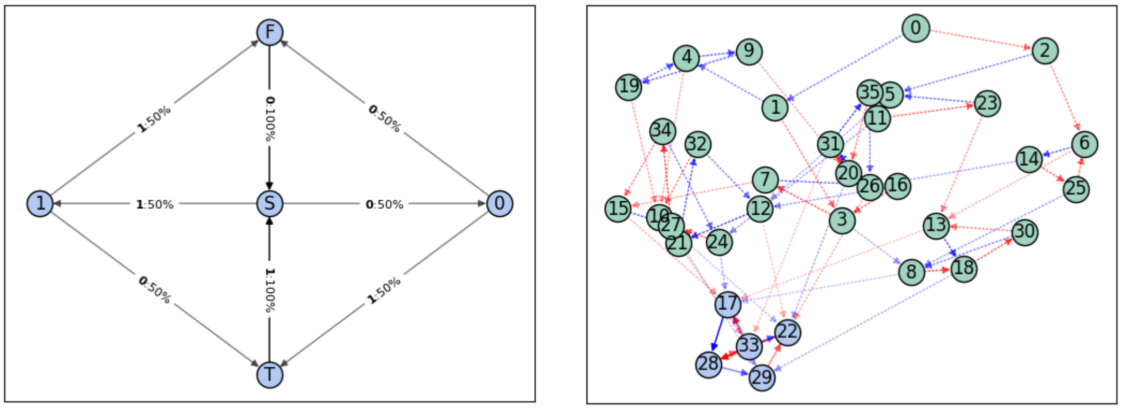

Figure 1. **a)** Representation of the RRXOR process. Nodes represent the hidden state of the HMM, while edges correspond to the probabilities of emission of certain tokens. **b)** Mixed state Presentation of the RRXOR process. Blue nodes are recurrent states, while green ones are transient. Red edges correspond to transitions where 1 is emitted, while blue edges correspond to transitions where 0 is emitted.

2.1.2 Visualizing the Mixed State Presentation 4-simplex

There are 436 unique 10 character sequences that can be generated by the RRXOR. We collect the activations of each layer over these 436 sequences. Future work could look at the activations of different components, attention and MLPs. At the end of this process, we have 436 residual streams * 10 tokens * 4 layers, which we can label with the correct belief state. We then do linear regression from the hidden dimension of the residual stream to the 4-simplex corresponding to the belief states. Because the belief state space is 5-d we then use a 3-d PCA to visualize both the activations and the ground truth representations of the belief states, see Figure 2, and see that the residual stream captures the expected geometry. The linear regression concatenates the activations of all layers, and so it goes from a 512 dimensional space to a 5 dimensional space (one for each state dimension).

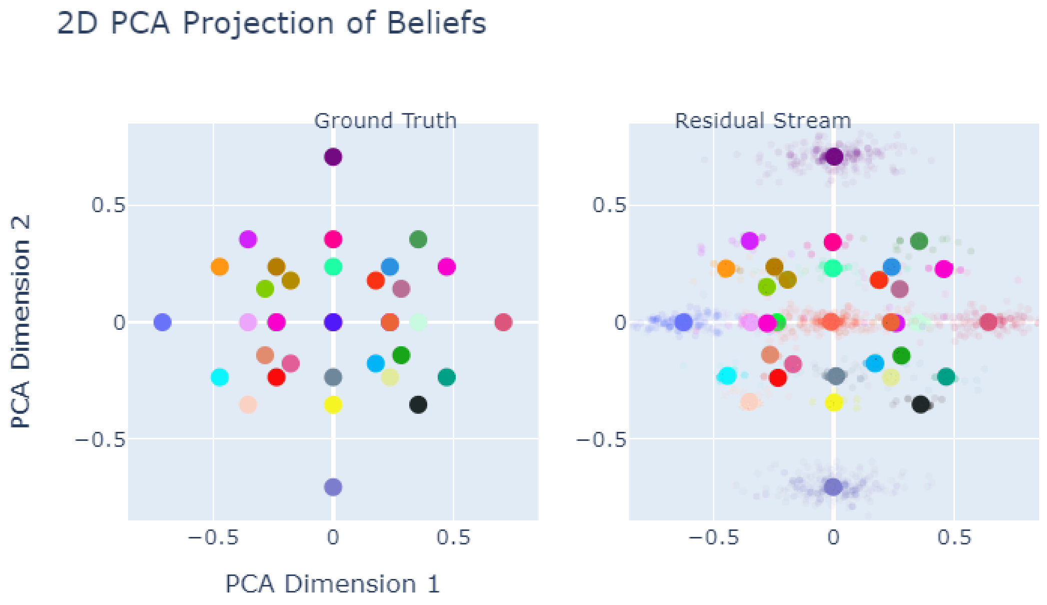

Figure 2. We use the first and third dimension of the PCA to represent the geometry of the belief state simplex. The left plot represents the ground truth labels, while the right one represents the projection of the residual stream after linear regression. Some states have a bigger cloud of points has these belief states are more common.

2.3 Computing steering vectors

We approach steering the activations of the model from one belief state to another in two different ways: 1) using the difference in means of activations and 2) using the belief state geometry.

Difference in means steering vectors

Having a labeled set of activations and associated belief states, we compute the mean activation for each belief state and compute the steering vector from belief state A to belief state B, $v_{A->B}$, as:

$$v_{A->B} = Act_B - Act_A,$$

Where Act_A and Act_B correspond to the average activations of each of the belief states. Each layer has its corresponding $v_{A->B}$.

To steer the model, $v_{A->B}$ is added to the residual stream at the token position where the belief state of the model should be A, and if steering works, the belief state of the model becomes B. We can choose to add a steering vector in all layers, the corresponding vector to each corresponding vector, or at individual layers, a single vector to its corresponding layer. We can also choose to add a single vector, e.g. the average of the vectors for all layers, to any single layer. These differences between these different approaches are discussed in the results section (see also Figure 3 and FigureS2).

Belief state steering vectors

One can also use the lower dimensional belief state geometry to compute the steering vectors. For example, if we want to steer from state F to state T, we would use their belief state representations, $State_F = (0,0,0,0,1)$ and $State_T = (0,0,0,1,0)$. If the X is the projection matrix corresponding to the linear regression from the residual stream to the belief state geometry, the steering vector is computed as:

$$v_{T->F} = X^{\dagger}(State_F \text{-} State_T)$$

Where $X^{\dagger}$ is the Moore-Penrose Pseudo-Inverse of X. We used these two states because the HMM always emits 0 in state F and 1 in state T (See Figure 1).

We applied this method in two slightly different ways. Once we computed these steering vectors using the regression on all layers and then splitting the vector back into 4 separate vectors for the residual stream in each of the 4 layers of the transformer. For the second variation, we computed a different regression for the activations in different layers and then used that regression to compute the steering vector for that layer.

2.3 Code and reproduction

The code to reproduce all our results can be found at: https://github.com/sonofhypnos/epsilon-transformers.

3. Results

Even though the distribution of possible belief states is imbalanced when considering sequences of length 10, with the recurrent states being very represented, we observe that the linear regression for the residual stream to the belief state geometry clearly identifies most states, as can be seen that the ground truth of the MSP is very similar to the projection of the residual stream (Figure 2, see also Figure S1).

We reason that if the Transformer is updating its beliefs in line with the MSP, we should be able to steer the model by two different methods. First, by shifting the belief state via the activations in the residual stream, we should be able to change the output of the model. Second, if the MSP is truly sufficient to describe the Transformer's behavior, we should be able to project onto the lower-dimensional 4-Simplex, and steer using only the projection.

3.1 Steering models outputs is possible by changing the models' belief state

We measure the effectiveness of our steering vectors by looking at the predicted emission of the Transformer. Because we know the probability distributions over tokens of each belief state, we can measure if steering the model into a different belief state results in a different probability distribution over tokens.

3.1.1 Using the difference in means of activations

We begin by steering via activations of the residual stream. We do this in each layer individually (L1-L4 in Figure 3 and FigureS2) as well as by averaging the activations across all layers. Interestingly, we find that steering using the last layer tends to work better than earlier layers (Figure S2 is more representative than Figure 3 in this regard). Additionally, steering using all the layers tends to mimic the results of steering via the last layer.

In Figure 3, we are trying to steer beliefs of State T (output = 1) towards State F (output = 0) and vice versa (see Figure 1a for HMM), which is most successful when steering with activations in L3, L4 or all layers.

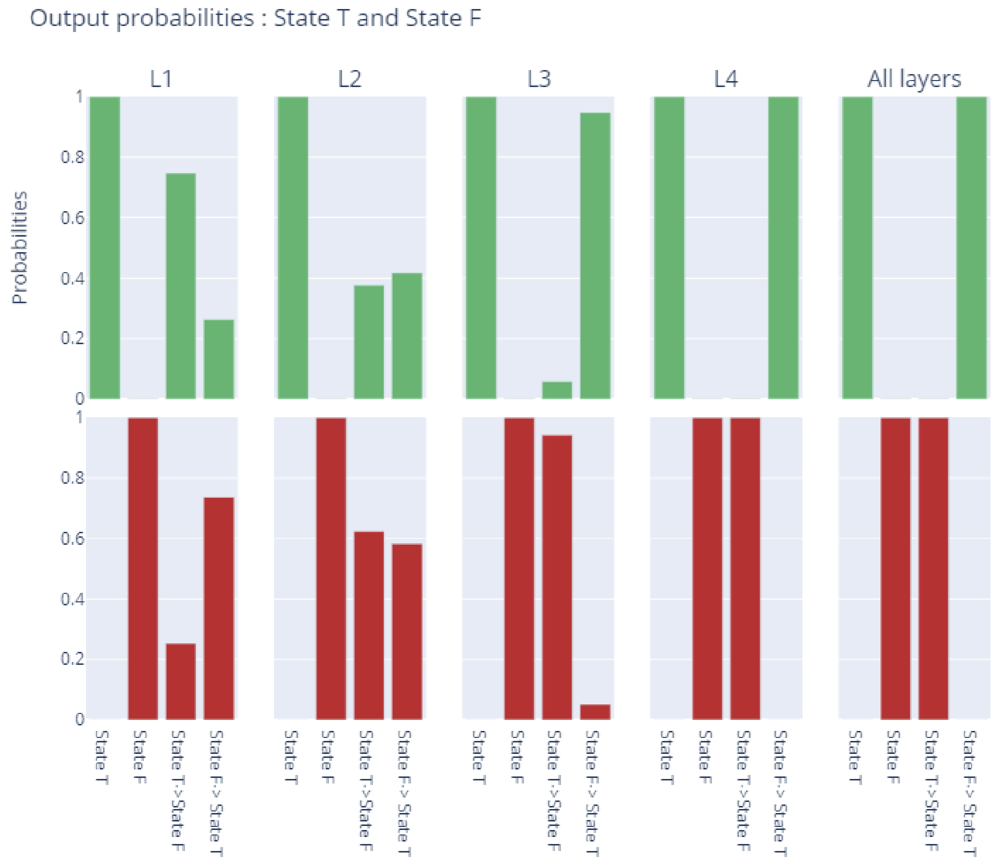

Figure 3. Output probabilities of States T and F before and after steering intervention. Steering vectors were either specific to a layer (L1-L4) or used activations from all layers (All layers). Green bars (top) indicate the probability of outputting 1. Red bars (bottom) indicate the probability of outputting 0

Although we can successfully steer the model, we observe that the steering vector obtained using the residual stream does not align perfectly with the difference between the steering centers when in the belief geometry state representation (Figure 4). This means that there maybe be some extra information that the model is using to define its belief states that is not captured in the belief state geometry. We also see that the steering vectors generated are longer than the distance between the belief states in the projection of the 4-simplex.

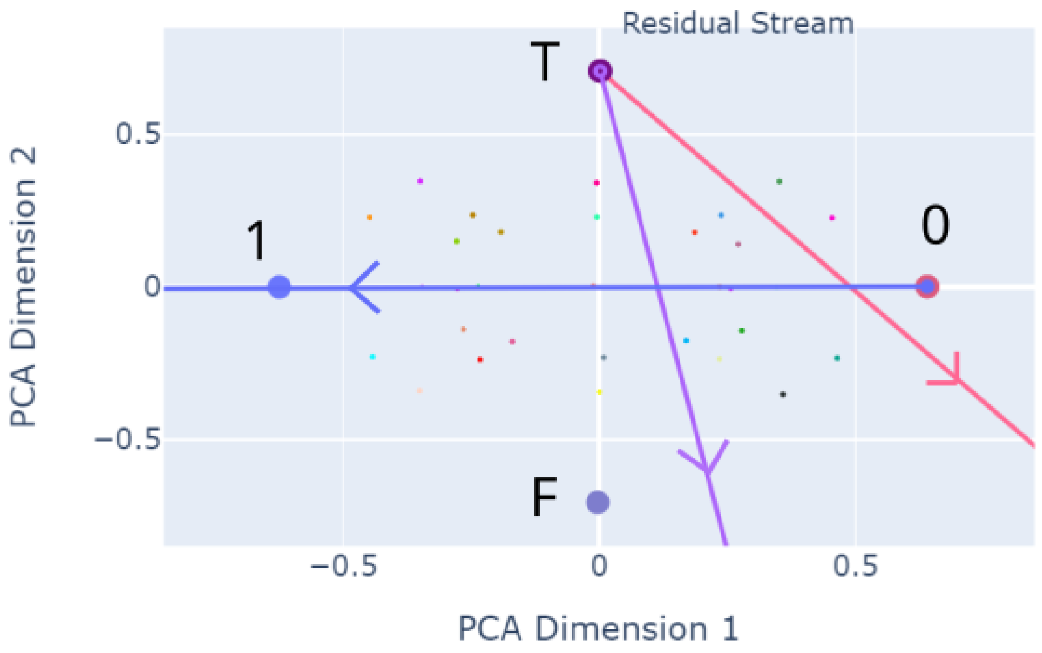

Figure 4. Steering direction in PCA space - The projected vectors that steer the model from belief state T to belief state 0 (pink) and from belief state T to belief state F (purple) and from belief state 0 to belief state 1.

3.1.2 Using the belief state geometry

If the belief state geometry truly represents the way the Transformer models its belief states, then we should be able to use the projected vectors from the 4-simplex obtained using linear regression and the original space of the activations of the model. This approach did not work - it didn't change the output probabilities at all! - and gave poor results even when using a multiplier, see Figure 5a).

We did successfully steer the model from to predict 1 instead of 0 or 0 instead of 1, which we interpret as successfully steering it from state T to F or from state F to T respectively, by doing a linear separate regression per layer. Using this method, we were able to steer the model when adding the respective steering vector to the residual stream in all the layers. Using per-layer steering vectors still worked in

>90% of cases when only intervening in the last two layers. For more details and other variations, see the `steering_rrxor_projection.ipynb` notebook.

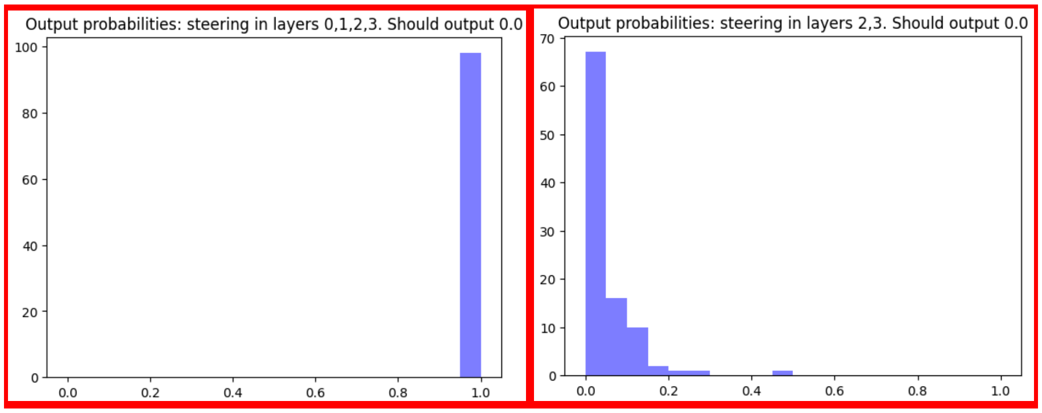

Figure 5. We choose a set of inputs corresponding to state T in the RRXOR process, which should make the model always thus predicts 1. **a)** Adding the steering vectors computed from the regression from the activations in the residual stream in all layers leaves the output completely unaffected. b) Using a steering vector in layers 2 and 3 (using a different separate regression for the projection in layers 2 and 3), the model is mostly predicting 0 as the next token.

4. Discussion and Conclusion

This project finds that steering can influence models' internal representations of the world, showing the potential of similar techniques to control more complex models' behaviors.

Despite the fact that the geometry of models' internal representation is predicted successfully by Computational Mechanics, we observe that it does not perfectly map to models' behaviors, Additionally, steering found by using the difference-in-means does not perfectly map to the belief state geometry,

Limitations:

Although this work shows that simple Transformers can create the full Mixed State Presentations when trained on simple Markov models, this approach is not what is happening in LLMs. Considering that Computatio1-nal Mechanics predictions still hold for current LLMs must mean that whatever Mixed State Presentation created by the LLM must be a coarse-grained one, and it is not so obvious how steering is changing the belief states.

Extensions:

There are several open questions: 1) Is there a better way to steer using the belief state geometry representation? 2) Can we track the updates between the belief states of the model and use them to find better steering vectors? 3) Are parts of the belief state distributed over the layers or is there redundant information?

5. References

[1]

A. Arditi, O. Obeso, Aaquib111, wesg, and N. Nanda, “Refusal in LLMs is mediated by a single direction,” Apr. 2024, Accessed: Jun. 02, 2024. [Online]. Available: https://www.lesswrong.com/posts/jGuXSZgv6qfdhMCuJ/refusal-in-llms-is-mediated-by-a-single-direction

[2]

N. Rimsky, N. Gabrieli, J. Schulz, M. Tong, E. Hubinger, and A. M. Turner, “Steering Llama 2 via Contrastive Activation Addition.” arXiv, Mar. 06, 2024. doi: 10.48550/arXiv.2312.06681.

[3]

Templeton, et al., "Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet", Transformer Circuits Thread, 2024.

[4]

A. Shai, “Transformers Represent Belief State Geometry in their Residual Stream,” Apr. 2024, Accessed: Jun. 02, 2024. [Online]. Available: https://www.lesswrong.com/posts/gTZ2SxesbHckJ3CkF/transformers-represent-belief-state-geometry-in-their

6. Appendix

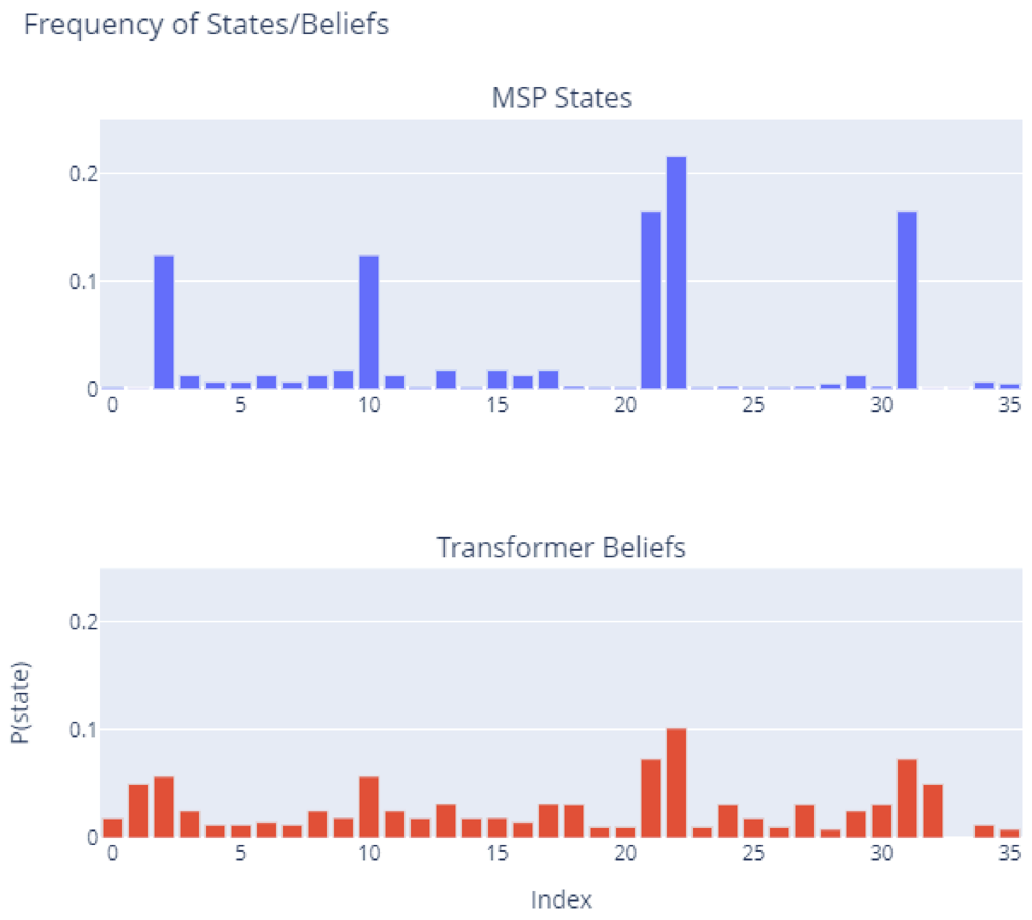

Figure S1. RRXOR process (top) and distribution of MSP state indices and Transformer input belief indices (bottom). In this figure, states [S, 0, 1, T, F] are associated with indices [22, 10, 2, 21, 31].

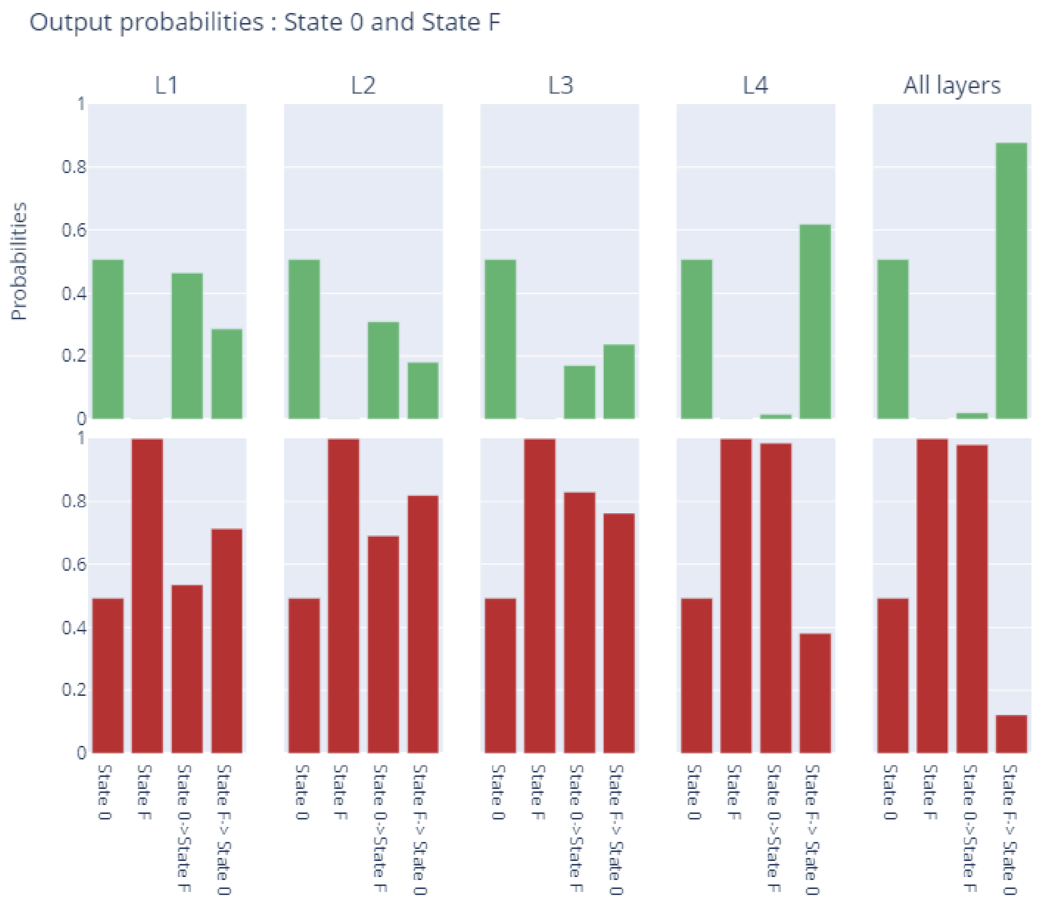

Figure S2. Output probabilities of States 0 and F before and after steering intervention. Steering vectors were either specific to a layer (L1-L4) or used activations from all layers (All layers). Green bars (top) indicate the probability of outputting 1. Red bars (bottom) indicate the probability of outputting 0.