Probe-Controlled TSV for Reducing Hallucinations in Language Models

December 3, 2025

1 Probe-Controlled TSV

`experiments/probe_controlled_tsv/`.

1.1 Step 1: Learn a Truthfulness Direction (TSV)

Given hidden states $\mathbf{h} \in \mathbb{R}^d$ from a fixed layer L^*, we want a direction vector $\mathbf{v}_{\text{TSV}} \in \mathbb{R}^d$ such that

- $\mathbf{v}_{\text{TSV}}^\top \mathbf{h}$ is large for *truthful* states,
- $\mathbf{v}_{\text{TSV}}^\top \mathbf{h}$ is small for *hallucinatory* states.

In the original TSV work [1], this is learned with an Optimal Transport objective. In our implementation, we use a simpler and more interpretable logistic regression classifier over hidden states; its weight vector becomes the TSV.

1.2 Step 2: Predict Hallucination Risk with a Probe

We train a small probe f_θ (MLP) that maps a hidden state $\mathbf{h}_t$ to a scalar risk score $\hat{r}_t \in [0, 1]$:

$$\hat{r}_t = f_\theta(\mathbf{h}_t), \quad \hat{r}_t \approx \Pr(\text{“about to hallucinate”} \mid \mathbf{h}_t).$$

The probe is trained on the same dataset and labels as TSV (BLEURT-bscore 0.5 threshold).

1.3 Step 3: Adaptive Steering During Generation

At each generation step t:

1. Run the LLM to obtain logits $\mathbf{z}_t \in \mathbb{R}^{|\mathcal{V}|}$ and hidden state $\mathbf{h}_t$ at layer L^*.
2. Compute risk $\hat{r}_t = f_\theta(\mathbf{h}_t)$.
3. Compute steering strength α_t:

$$\alpha_t = \begin{cases} 0, & \hat{r}_t < \tau, \\ \alpha_{\max} \cdot \dfrac{\hat{r}_t - \tau}{1 - \tau}, & \hat{r}_t \geq \tau, \end{cases}$$

   where τ is a risk threshold.

4. Apply steering in logit space using TSV .

This realizes the “only when needed” behavior: low risk ⇒ no intervention; high risk ⇒ stronger steering.

2 Dataset and Label Construction

We follow TSV's two-stage pipeline on TruthfulQA:

1. **Answer generation:** For each TruthfulQA question q_i, the base model (GPT-Neo-1.3B) generates one or more answers, saved under `save_for_eval/tqa_hal_det/answers/*.npy`.
2. **BLEURT scoring:** A BLEURT-like score $s_i \in [0, 1]$ is computed for each answer with respect to ground-truth references, stored in `ml_tqa_bleurt_score.npy`.

The sample generation follows the original tsv code
Total samples: 817 (TruthfulQA validation).

3 Logistic Regression TSV

3.1 traing data

Let x_i be the corresponding tokenized input. We run the model with hidden states output enabled:

$$\text{outputs} = \text{LM}(x_i; \texttt{output\_hidden\_states=True}),$$

and extract hidden states from layer L^* (e.g., layer 9) for the last non-padding token:

$$\mathbf{h}_i \in \mathbb{R}^d = \text{hidden\_states}[L^*][\text{batch}, t_{\text{last}}, :].$$

This yields a feature matrix $X \in \mathbb{R}^{N \times d}$ ($N = 817$) and label vector $y \in \{0, 1\}^N$.

3.2 Logistic Regression Formulation

We fit a binary logistic regression:

$$P(y = 1 \mid \mathbf{h}) = \sigma(\mathbf{w}^\top \mathbf{h} + b),$$

The optimization problem is:

$$\min_{\mathbf{w},b} -\sum_{i=1}^{N} \Big[y_i \log \sigma(\mathbf{w}^\top \mathbf{h}_i + b) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^\top \mathbf{h}_i + b)) \Big] + \lambda \|\mathbf{w}\|_2^2,$$

with $\lambda = 1/C$ the regularization strength.

Implementation :

```
clf = LogisticRegression(
    C=C, max_iter=1000, solver='lbfgs', class_weight='balanced'
)
clf.fit(X_train, y_train)

w = clf.coef_[0]  # [d]
b = clf.intercept_[0]
```

TSV definition. We define the TSV direction as

$$\mathbf{v}_{\text{TSV}} = \mathbf{w},$$

which is the normal vector of the decision hyperplane $\mathbf{w}^\top \mathbf{h} + b = 0$. By construction, $\mathbf{w}$ points from class 0 (hallucinations) towards class 1 (truthful states).

Geometrically, the decision boundary separates the two clusters, and $\mathbf{v}_{\text{TSV}}$ points in the direction that increases the log-odds of being truthful:

$$\log \frac{P(y=1|\mathbf{h})}{P(y=0|\mathbf{h})} = \mathbf{w}^\top \mathbf{h} + b.$$

3.3 Layer-wise TSV Storage

We train TSV on a single layer L^* (e.g., 9), but save it in a layer-wise list:

$$\texttt{tsv\_vectors} = [\mathbf{0}, \ldots, \mathbf{0}, \mathbf{v}_{\text{TSV}}, \mathbf{0}, \ldots, \mathbf{0}],$$

so that other components can index the correct layer by ID.

The checkpoint is saved as:

- `tsv_single`: the single TSV vector,
- `tsv_vectors`: list over all layers,
- training and test AUC/accuracy,
- model name, layer ID, threshold, etc.

4 lm_head Constraint: Preventing Logit Explosion

4.1 The Problem

If we directly steer hidden states by

$$\tilde{\mathbf{h}} = \mathbf{h} + \alpha \mathbf{v}_{\text{TSV}},$$

the logits change by:

$$\Delta \mathbf{z} = \alpha \mathbf{v}_{\text{TSV}}^\top W_{\text{lm}},$$

where $W_{\text{lm}} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the `lm_head` weight matrix, and $\mathbf{z} = W_{\text{lm}} \mathbf{h}$.

In practice, we observed extreme logit changes (+472 for a single token), leading to:

- Probability mass collapsing onto a single token,
- repeating a weird subword.

4.2 Constraint Formulation

We introduce a hard constraint on the maximum allowed logit change for a given steering strength α:

$$\max_{v \in \mathcal{V}} |\Delta z_v| = \max_{v \in \mathcal{V}} \left| \alpha \, \mathbf{v}_{\text{TSV}}^\top W_{\text{lm}}^{(v)} \right| \leq \Delta_{\max},$$

where $W_{\text{lm}}^{(v)}$ is the row of W_{lm} for token v, and $\Delta_{\max}$ is a small constant (e.g., 3.0).

We enforce this post-hoc by scaling $\mathbf{v}_{\text{TSV}}$:

$$s = \frac{\Delta_{\max}}{\max_v |\alpha\, \mathbf{v}_{\text{TSV}}^\top W_{\text{lm}}^{(v)}|}, \qquad \mathbf{v}_{\text{TSV}} \leftarrow \begin{cases} s\, \mathbf{v}_{\text{TSV}}, & s < 1, \\ \mathbf{v}_{\text{TSV}}, & s \geq 1. \end{cases}$$

Implementation:

```
delta_logits = alpha * tsv_vector @ lm_head_weight.T
max_change = np.abs(delta_logits).max()
if max_change > max_logit_change:
    scale = max_logit_change / max_change
    tsv_vector = tsv_vector * scale
```

After this constraint, we empirically reduce max logit change from ~ 472 to ~ 1.2, preventing catastrophic steering.

4.3 Logits-Space Steering

Rather than modifying hidden states directly, we steer in logit space:

$$\tilde{\mathbf{z}}_t = \mathbf{z}_t + \alpha_t\, \mathbf{s},$$

where

$$\mathbf{s} = \mathbf{v}_{\text{TSV}}^\top W_{\text{lm}} \in \mathbb{R}^{|\mathcal{V}|}$$

is the TSV-induced shift in logits.

In practice, for numerical stability we:

- Precompute $\mathbf{s}$ once per experiment,
- Apply $\tilde{\mathbf{z}}_t = \mathbf{z}_t + \alpha_t\, \mathbf{s}$ only when $\alpha_t > 0$.

5 Probe Architecture and Training

We adopt a simple MLP probe:

$$f_\theta(\mathbf{h}) = \sigma\left(\mathbf{w}_2^\top \phi(\mathbf{W}_1 \mathbf{h} + \mathbf{b}_1) + b_2\right),$$

where: $\mathbf{W}_1 \in \mathbb{R}^{m \times d}$, $m = 256$, ϕ is GELU, σ is sigmoid.

Given the same features $\mathbf{h}_i$ and labels y_i as TSV, we optimize:

$$\mathcal{L}_{\text{probe}} = -\sum_i \left[y_i \log f_\theta(\mathbf{h}_i) + (1 - y_i) \log(1 - f_\theta(\mathbf{h}_i))\right].$$

We report train/test AUC; in our experiments:

- Train AUC ≈ 0.92,
- Test AUC ≈ 0.80.

This indicates the probe can reliably distinguish high- vs low-risk states.

6 Adaptive Steering Dynamics

At generation step t, the procedure is:

1. Compute hidden state $\mathbf{h}_t$ at layer L^*.
2. Risk prediction:
$$\hat{r}_t = f_\theta(\mathbf{h}_t) \in [0, 1].$$
3. Steering strength:
$$\alpha_t = \begin{cases} 0, & \hat{r}_t < \tau, \\ \alpha_{\max} \cdot \dfrac{\hat{r}_t - \tau}{1 - \tau}, & \hat{r}_t \geq \tau. \end{cases}$$
4. Logit update:
$$\tilde{\mathbf{z}}_t = \mathbf{z}_t + \alpha_t \, \mathbf{s}, \quad \mathbf{s} = \mathbf{v}_{\mathrm{TSV}}^\top W_{\mathrm{lm}}.$$
5. Sample next token using $\tilde{\mathbf{z}}_t$ with top-p sampling.

7 Experimental Setup

- **Model:** GPT-Neo-1.3B (EleutherAI).
- **Dataset:** TruthfulQA generation split, 817 validation questions.
- **Layer:** $L^* = 9$ (empirically selected).

We evaluate three conditions:

- **Baseline:** No steering ($\alpha_t = 0$).
- **Fixed:** Fixed $\alpha_t = \alpha_{\text{fixed}}$ for all tokens.
- **Adaptive (Ours):** Probe-controlled α_t as described above.

7.1 Metrics

- **Accuracy:** 1− hallucination rate, where hallucination is defined by substring match against references.
- **Hallucination rate:** fraction of outputs not containing any reference answer.
- **BLEURT-like score:** heuristic semantic similarity between output and references.
- **Style similarity:** cosine similarity between last-layer embeddings of model outputs and reference-style templates.
- **Steering rate:** fraction of tokens with $\alpha_t > 0$.

8 Results

8.1 Main Comparison

On 20 sampled TruthfulQA questions:

Method	Accuracy	Hal Rate	BLEURT	Style Sim
Baseline	0.25	0.75	0.321	0.818
Fixed	0.25	0.75	0.308	0.788
Adaptive	**0.40**	**0.60**	**0.405**	0.808

Adaptive steering improves accuracy and BLEURT.

8.2 TSV and Probe Quality

- TSV logistic regression:
  - Train AUC: 0.9947,
  - Test AUC: 0.7696.
- Probe MLP:
  - Train AUC: 0.9203,
  - Test AUC: 0.8047.

These values indicate that both TSV direction and probe risk scores capture meaningful information about truthfulness.

9 Discussion and Innovation

10 Conclusion and Future Work

We demonstrated that a Probe-Controlled TSV approach can reduce hallucinations on TruthfulQA while preserving style and fluency, using a relatively small model (GPT-Neo-1.3B) and modest computational resources.

Future directions:

- Scale to larger models (LLaMA-2, Mistral) and datasets (FEVER, HaluEval).
- Use Tuned-Lens-style features as probe inputs to capture layer-wise trajectories.
- Explore multi-layer TSVs and richer control policies over α_t.

References

[1] Park et al. Steer LLM Latents for Hallucination Detection. 2025.

[2] Wang et al. Adaptive Activation Steering (ACT). 2024/2025.

[3] Rimsky et al. Contrastive Activation Addition. 2024.

[4] Belrose et al. Tuned Lens: Eliciting Latent Predictions from Language Models. 2023.