Detecting LLM Hallucinations Through Probability Geometry

David Backer Peral, Helen Mao, Max Lu, Ethan Hao, Kevin Guo
The Glittering Pink Fluffy Rainbow Bunny Brigade
AMTH 232 — Advanced Linear Algebra with Applications, Yale University

***Abstract*—Large language models often produce confidently wrong answers. In this project, we investigate whether the shape of the model's own next-token probability distributions can tell us, without ever looking at the answer, whether the response is going to be a hallucination. We run Qwen2.5-1.5B-Instruct on 300 training GSM8K math problems and 300 training CommonsenseQA (CSQA) multiple choice questions, save the top-1000 probabilities at every generation step, and compress each response into a small 9-dimensional feature vector built from three per-step quantities (top-1 probability, Gini coefficient, and perplexity) summarized by three aggregates (mean, standard deviation, last value). On these vectors we apply four main analyses: Principal Component Analysis (PCA), Laplacian Eigenmaps, Logistic Regression, and neural networks. We evaluated the results of the model on 300 new problems on each of the datasets. The hypothesis, that flatter token distributions correlate with hallucinations, is supported, especially on GSM8K. The strongest detector on GSM8K is Logistic Regression (AUROC 0.714), and the strongest on CSQA is Laplacian Eigenmaps (AUROC 0.631). However, our results indicate that a classifier trained on one dataset does not transfer to the other, suggesting that our results are not generalizable across datasets.**

I. Introduction

In the last several years, Large Language Models (LLMs) have become a central part of cursory research and critical operations across many fields, from academia to industry. However, LLMs are still not fully reliable. Although they improve with every iteration, we remember when it was easy to convince ChatGPT that $2 + 3 = 6$. Some wrong answers are inevitable, but can we detect them?

Frequent LLM users have likely observed a pattern of LLMs confidently outputting an incorrect answer as in the example above—a phenomenon known as 'hallucination'. Detecting hallucinations is critical to interpreting output and understanding the powers and limitations of LLMs. It is currently difficult to place full trust in a LLM response due to their opaque sourcing and reasoning processes.

This unreliability motivates our two primary research questions:
1) Can we predict when an LLM hallucinates using the probabilistic distribution of tokens at each time step?
2) If so, can we determine which features are correlated with hallucinations and formulate an intuitive explanation?

The key observation of this project is that LLMs are nondeterministic (i.e. different responses produce different outputs). At each time step, it produces a probability vector p_t across its entire vocabulary (about $152{,}000$ words for the model we use) and then selects the most likely token. That probability vector is normally thrown away the moment the next token is sampled but we keep it for our analysis.

Our hypothesis is that when the model is certain of the answer, the probability distribution is sharply peaked, meaning that nearly all the mass is distributed to one word. As a corollary, when the model is unsure, we expect the distribution to flatten as many candidate tokens compete for the next slot. So if the trajectory $p_1, p_2, \ldots, p_T$ contains several flat steps, we expect the response to be a hallucination.

We test this by turning each response into a small fixed-size feature vector that captures how peaked or flat its distributions are (using statistics based on entropy). We apply four different linear algebra and general machine learning techniques: Principal Component Analysis (PCA), Laplacian Eigenmaps, Logistic Regression, and neural networks. We ran our analyses on two datasets to compare reproducibility different domains: GSM8K and CommonsenseQA (CSQA).

II. Background

A. How an LLM picks the next token

An LLM selects tokens by processing an input—consisting of the prompt and tokens it has already processed—and generating a probability distribution across its entire vocabulary. The LLM output is simply a collection of tokens that correspond to the highest probability within the distribution at each time step. As such, we can think of each token as the highest-probability prediction based on the given context.

B. Measuring how peaked a distribution is

The shape of the distribution can be quantified using the Shannon entropy, which is given by

$$H(p) = -\sum_{v} p(v) \log p(v) \tag{1}$$

As established, a confident, sharply peaked distribution results in lower entropy, whereas a flat, uncertain distribution results in higher entropy. We can utilize entropy to calculate the perplexity, which is defined as $\mathrm{PP}(p) = e^{H(p)}$. We interpret perplexity as the effective number of equally likely tokens. According to our hypothesis a confident answer should have low perplexity, whereas a hallucination would likely have higher perplexity. We also use the Gini coefficient $G(p)$, a measure of inequality across the distribution. It is high when one token has almost all the mass, and low when the mass

is distributed evenly. Our third metric is the top-1 probability $p^{(1)} = \max_v p(v)$, which represents the probability assigned to the output token. We extract these features when compressing each response into a multi-dimensional feature vector.

C. Datasets

GSM8K [1] is a set of grade-school math word problems with a single numeric answer per problem. CommonsenseQA (CSQA) [2] is a dataset that tests common sense reasoning through multiple-choice questions with five options. Both datasets have objective "correct" answers, which makes determining hallucinations easy.

D. Defining a hallucination

While there are many ways to define a hallucination, we utilize the simplest metric that allows for automatic labeling: a response is a hallucination if it yields the wrong final answer. This objective approach allows us to simplify the analysis such that we can ignore the model's chain of reasoning and instead just consider the final answer.

III. Methods

A. Qwen Model and Decoding

We use Qwen2.5-1.5B-Instruct [3], a 1.5B-parameter open-source model released by Alibaba Cloud. We load it in float16 on Apple's MPS backend, which fits in $3\,$GB of memory. During generation, when choosing the next token, we utilize greedy decoding with max_new_tokens=256. For each question, we prompt the model to output "The answer is: X" and evaluate its performance with a regex on the responses.

B. Datasets and splits

For each dataset, we randomize the questions and split them into a train and evaluation split. The model does not train on the evaluation data until the evaluation step. The first 300 questions are chosen for training, and the next 300 become the mutually exclusive evaluation questions.

C. Distribution preprocessing

We keep only the top-1000 probabilities, their corresponding tokens, and a single residual $r_t = 1 - \sum_{i \le 1000} p^{(i)}$. In practice, r_t is usually small and contributes very little to entropy, so we proceed as if it were negligible. We do this instead of saving the softmax over the full vocabulary at every step because the latter would require too much storage.

D. Features per step

At each step t, we determine three values from p_t, the output distribution at time t:

- **top-1**, the largest probability $p_t^{(1)}$;
- **Gini**, the Gini coefficient $G(p_t)$;
- **perplexity**, $\mathrm{PP}(p_t) = e^{H(p_t)}$.

E. Aggregating features per response

A response of length T tokens yields a $T \times 3$ matrix of data. We then collapse it to a fixed-size summary by computing three aggregates for each feature: the mean over all steps, the standard deviation over all steps, and the last value. Thus, we calculate 9 aggregates per response. We chose this 9-dimensional setup over a 65-dimensional one we tried in a previous version of our code because the reduced version was cleaner to interpret. The classifiers we chose were simpler, and it performed comparably on the evaluation, indicating that the other aggregates seemed to mainly just introduce noise. The specific features and aggregators selected were chosen through our analysis as well.

F. Evaluation

Our main metric is Area Under the ROC curve (AUROC), which measures the probability that a classifier scores a positive result higher than a negative one. In the context of this study, a positive result is a correct response and a negative is a hallucination. If it classifies both classes perfectly, AUROC = 1. If it guesses randomly, AUROC = 0.5. This means that AUROC is robust to datasets with unequally distributed categories, which was exactly what we needed since the model is right more often than wrong. In our tests, we saw that Qwen achieved 63% accuracy on the GSM8K benchmark and 72% accuracy on CSQA. Thus, we determine that the respective hallucination rates are 37% and 28%.

G. Classifiers

On the 9-dimensional vectors we run four methods, all from AMTH 2320:

Principal Component Analysis. We standardize the 9-dimensional vectors and extract the top 2 principal components PC1 and PC2. Each response's vector is projected onto these principal components, and the signs of each component are chosen so higher values correspond to non-hallucination responses. Then, we perform a logistic regression on the two values to synthesize the information into a single probability estimate representing the probability of a hallucination. All three—PC1 alone, PC2 alone, and their logistic combination—are then evaluated via AUROC.

Laplacian Eigenmaps. We build a Gaussian-kernel similarity matrix $W_{ij} = \exp(-\|x_i - x_j\|^2/2\varepsilon)$ with ε set to the squared median pairwise distance, form the normalised Laplacian $L_{\mathrm{sym}} = I - D^{-1/2} W D^{-1/2}$, and embed each data point using the graph's second and third eigenvectors, as the first is trivial. Then, this two-dimensional embedding $Z = [v_2, v_3]$ is used for binary classification in 3 ways.

- **1. and 2. Both eigenvectors are treated as a score.** The sign of the Fiedler vector v_2 tends to separate the two classes, and we flip it if necessary so that higher values correspond to correct predictions by the model.
- **3. A logistic regression is fit on** Z, synthesizing the information from both eigenvectors into a single probability estimate.

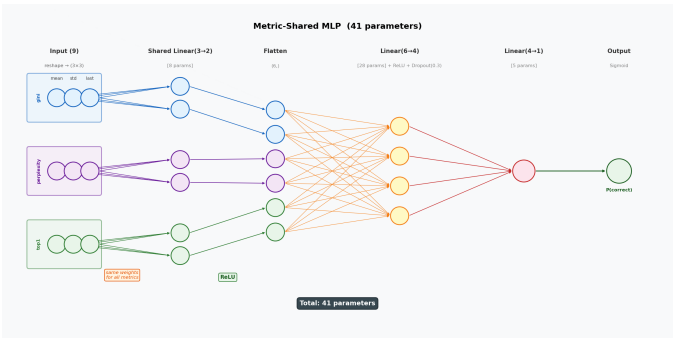


Fig. 1. MLP Architecture

All three classification models—v_2, v_3, and their logistic combination—are evaluated separately using AUROC and compared against each other.

Logistic Regression. A standard L2-regularized logistic regression model with $C = 1$ on the 9 standardized features. This is especially interpretable, as the trained parameters give us a sign and magnitude for each feature's impact on the final estimated probability.

Neural Networks. We utilized a small MLP structure with only 41 parameters. The 9-dimensional input is reshaped as a 3×3 matrix. A single shared $\text{Linear}(3 \to 2)$ is applied to each metric row, the output is flattened to $\mathbb{R}^6$, and a small dense network ($\text{Linear}(6 \to 4) \to \text{ReLU} \to \text{Dropout}(0.3) \to \text{Linear}(4 \to 1) \to \text{Sigmoid}$) produces the final probability. The MLP utilizes early stopping (patience=30, up to 500 epochs), an Adam optimizer with $lr = 1e{-}3$, $weight_decay = 1e{-}3$, and 5-fold cross-validation on the training set. A visual representation of the architecture can be seen in Figure 1.

IV. Results

A. Are the features actually different between correct and hallucinated responses?

Before fitting any classifier, it is worth checking whether the features themselves separate the two classes at all. Figure 2 overlays the per-class histograms of mean perplexity, mean Gini, and mean top-1 for both datasets.

On GSM8K the means clearly differ. Hallucinations have a higher mean perplexity (2.48 vs 2.05) and a lower mean Gini and a lower mean top-1, in line with our hypothesis. On CSQA, the histograms overlap much more. This can be attributed to the short, standardized 6-token length of CSQA responses, a smaller sample size for summary statistics. This could hint at the predictive accuracy of the models trained on the two different datasets.

B. PCA

Figure 3 shows the 2-D PCA projections, the loadings of PC1 and PC2, and the resulting ROC curves. On GSM8K, PC1 captures 62.5% of the variance and is dominated by Gini-related and perplexity-related features—it scores AUROC 0.682 on its own. PC2 adds 26% of variance but only 0.578 AUROC, so combining the two through a small logistic regression gives 0.691 AUROC, only marginally better than PC1 alone.

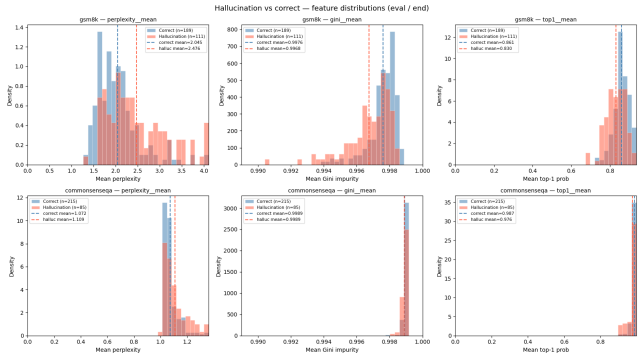


Fig. 2. Per-class histograms of the three feature means. On GSM8K (top row) hallucinations sit at noticeably higher mean perplexity and slightly lower mean Gini and top-1, exactly as the hypothesis predicts. On CSQA (bottom row) the gap is much smaller because responses are very short (6 tokens median).

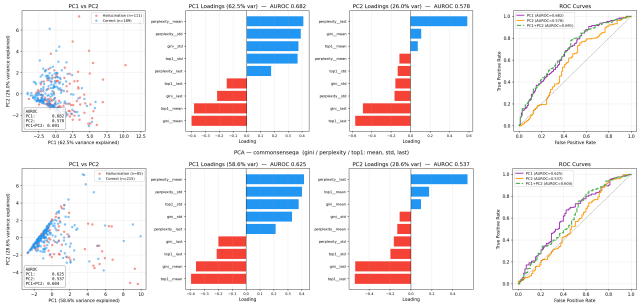


Fig. 3. PCA results on GSM8K (top) and CSQA (bottom). Each panel shows the PC1-vs-PC2 scatter (correct in blue, hallucinations in red), the PC1 loadings, the PC2 loadings, and the ROC curves obtained by scoring with PC1, PC2, and PC1+PC2.

The CSQA picture is structurally similar but weaker: PC1 yields AUROC 0.625, PC2 only 0.537, and the combined score reaches 0.604. The dominant directions in feature space line up with our intuition: the projection is essentially picking up "flatness on average and at the last step," but the classes overlap more.

C. Laplacian Eigenmaps

Figure 4 shows the results of applying Laplacian Eigenmaps. The embeddings on both datasets are dominated by a small number of outliers, but the signal is still there: on GSM8K LE1 alone reaches AUROC 0.679, and combining LE1 with LE2 gives 0.686. Crucially, the correlations per-feature show *that the Gini* features (gini_last, gini_mean) are more strongly correlated with the discriminating direction.

Applying Laplacian Eigenmaps only produced stronger results than simpler methods on the CSQA dataset. They achieved AUROC 0.631, slightly higher than logistic regression (0.616). We theorize that Laplacian Eigenmaps performed better here because the short CSQA responses require non-linear models to find the separation.

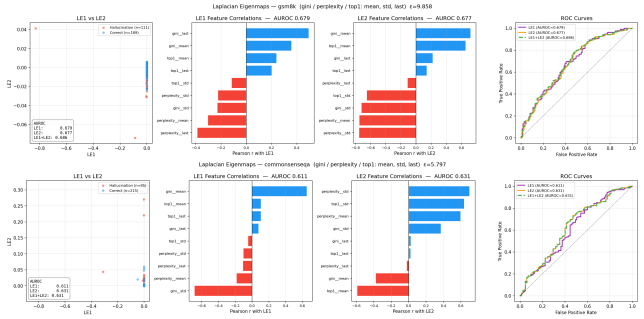


Fig. 4. Laplacian Eigenmaps on GSM8K (top) and CSQA (bottom). The 2-D embeddings are dominated by a few outliers, but the correlation panels show that Gini-based features are the strongest contributors to the discriminating direction in both datasets.

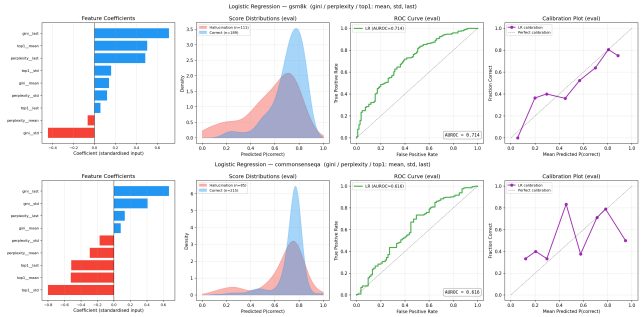


Fig. 5. Logistic regression coefficients, score distributions, ROC curve and calibration plot on GSM8K (top) and CSQA (bottom). The largest GSM8K coefficient is on gini_last.

D. Logistic Regression

Logistic regression on the same 9 standardized features is the best GSM8K classifier we have (Fig. 5, AUROC 0.714). We interpret the coefficient bar chart where the strongest positive coefficient is on gini_last and the largest negative coefficient is on gini_std. In words: a high Gini at the very last step is a strong correctness signal (the model finished confident), while a spread out Gini across the response (the response had at least one flat distribution somewhere) is a hallucination signal.

The score distribution panel shows the two classes overlap heavily, suggesting that the model provides moderate, not strong, discrimination. The calibration plot on GSM8K is reasonably close to the diagonal, so the predicted probabilities are viable as confidence scores. Conversely, the CSQA are poorly calibrated, again likely due to the dataset's short response lengths.

E. Neural Networks

Figure 6 shows the small MLP. On GSM8K it reaches AUROC 0.698, slightly worse than the linear logistic regression but comparable. On CSQA it underperforms (0.588). The shared-layer heatmap reveals that the network mostly learned to weight the "std" aggregate negatively and "mean" positively—which aligns with the findings from logistic regression.

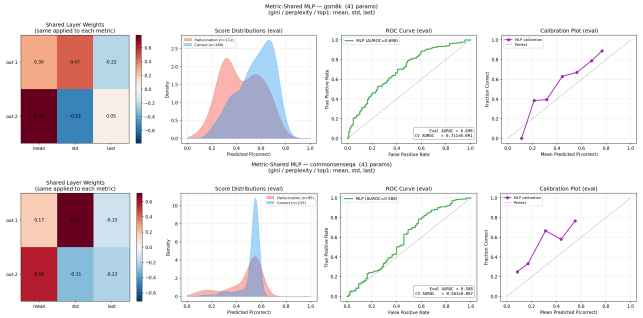


Fig. 6. Metric-shared MLP results on GSM8K (top) and CSQA (bottom). The leftmost panel is the heatmap of the 2×3 shared weight matrix.

TABLE I
HELD-OUT EVALUATION AUROC FOR THE FOUR CLASSIFIERS ON BOTH DATASETS.

Method	GSM8K	CommonsenseQA
PCA (PC1+PC2)	0.691	0.604
Laplacian Eigenmaps	0.686	**0.631**
Logistic Regression	**0.714**	0.616
Metric-Shared MLP	0.698	0.588

Our PCA results explain why a 41-parameter network cannot beat a logistic regression on just 300 training examples: most of the signal lives in a low-dimensional, roughly linear subspace. The data simply lacks the non-linear complexity for a small MLP to exploit.

F. Putting it all together

Table I collects the headline AUROC numbers.

G. Cross-dataset transfer

We next followed up with a question asking whether the geometric signal is global or domain-specific: if we train on one dataset, does the classifier still work on the other one? Figure 7 answers this. The "same dataset" bars are the AUROCs we already reported. The "cross dataset" bars are what happens when we train on GSM8K and test on CSQA, or vice versa. Both Logistic Regression and the MLP collapse sharply in the GSM8K $\rightarrow$ CSQA direction (AUROC drops below 0.5 in some cases, meaning the classifier is actively wrong). Going CSQA $\rightarrow$ GSM8K is less catastrophic but still well below the same-dataset number.

We interpret these results as generally supportive of our hypothesis. While the probability geometry consistently predicts hallucinations within a single dataset, the decision thresholds are strictly domain-specific. Math responses and one-letter multiple-choice responses produce probability trajectories with very different scales of perplexity and Gini, and a classifier tuned for one scale does not carry over.

V. DISCUSSION

A. The signal is mostly linear

The AUROC scores support our hypothesis that token probability geometry can detect hallucinations. As we anticipated, hallucinated responses consistently exhibited flatter

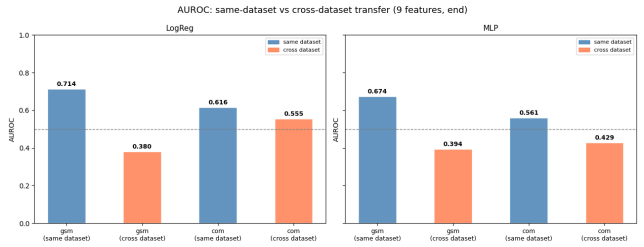


Fig. 7. Same-dataset vs cross-dataset AUROC. Training on GSM8K and testing on CSQA (or vice versa) loses most of the signal — the geometric structure exists in both domains but does not naively transfer.

distributions, characterized by higher entropy, and Gini was the most dominant feature in logistic regression. Simple linear models performed best on GSM8K, while PCA and Laplacian Eigemaps captured the vast majority of variance in just their first few components. Thus, we conclude that the geometric signal separating real answers from hallucinations is primarily linear and low-dimensional.

B. Why CSQA is harder

The models score consistently lower on CSQA, likely due to the brief 6-token answer format ('The answer is: X'). Such short responses generate too few data points to reliably measure entropy or Gini inequality, whereas GSM8K's longer responses (more tokens) provide a richer probability trajectory to analyze. Interestingly, the Laplacian Eigenmap works best on CSQA, perhaps finding more non-linear local geometry than a strong linear signal that PCA can capture.

C. Strength of features

Although each classification method has different features that take the most precedence, Gini appears most often as the strongest indicator of hallucinations. In particular, the Logistic regression's biggest coefficient is with Gini of the last token distribution, and Laplacian Eigenmap correlations are strongest for Gini features. The limited predictive power of the top-1 probability is logical, given that it isolates a single value and ignores the rest of the distribution. What is more surprising, however, is that Gini outperformed perplexity as the primary discriminative measure of output uncertainty.

D. Limitations

Our results are AUROCs in the 0.6-0.7 range, indicating only modest accuracy, but they successfully demonstrate a meaningful geometric relationship between entropy measures and hallucinations. Future research should expand the size of the dataset to support more complex models without overfitting and perform formal hypothesis testing to establish the statistical significance of these AUROC results. Furthermore, standardizing the prompting strategy across datasets could effectively control for extreme variations in response length. Finally, overcoming our local hardware constraints would allow us to scale this probability analysis to much larger language models that are more applicable to what people actually use in day to day life.

VI. Conclusions

Using linear algebra tools from class, we tested whether the geometry of an LLM's probability distributions could predict hallucination. The signal is effective on GSM8K and weaker but still detectable on CSQA. Gini-based features on average had the highest predictive power. Overall, their signal is well captured with a linear model. More complex models tend to capture more local structure and perform poorly cross-datasets. The breakdown of PCA and Laplacian Eigenmap components gives intuition for the geometric structure of the data, helping us understand the probability tokens' predictive power. Ultimately, our results demonstrate that the entropy and flatness of token probability distributions can identify hallucinations. However, the predictive accuracy is moderate and classifiers trained on one domain do not easily generalize to others. Despite the need for domain-specific calibration, the underlying geometric signal remains robust. Rather than acting as a standalone solution, our findings suggest that entropy and distributional inequality can be combined with other verification methods to build more robust hallucination detectors. This quantitative approach to uncertainty provides a valuable mechanism for mitigating the risks of AI misinformation.

Appendix

The full source code for this project, including data preprocessing, feature extraction, and all classifier implementations, is available at:

https://github.com/Davimeleon/LLM-Hallucinations-AMTH-232.git

References

[1] K. Cobbe *et al.*, "Training Verifiers to Solve Math Word Problems," *arXiv:2110.14168*, 2021.
[2] A. Talmor, J. Herzig, N. Lourie, and J. Berant, "CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge," *Proc. NAACL*, 2019.
[3] Qwen Team, "Qwen2.5 Technical Report," Alibaba, 2024.
[4] S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal, "Detecting hallucinations in large language models using semantic entropy," *Nature*, vol. 630, pp. 625–630, 2024.
[5] Z. Ji *et al.*, "Survey of Hallucination in Natural Language Generation," *ACM Computing Surveys*, vol. 55, no. 12, 2023.