

Hallucinations in Large Language Models

By
Lokesh RLN – 622246
Ch Mokshagna – 622129
Dinesh K – 622171

Under Supervision of P. Usha



National Institute of Technology Andhra Pradesh

Introduction: What is hallucination?

- **Definition:** Hallucination in large language models (LLMs) refers to confidently produced statements that are factually incorrect, fabricated, or unsupported by the provided input.
- **Why it happens:** Models optimize for plausible continuations of text, not factual correctness; training data noise, distribution shifts, and insufficient grounding can lead to made-up answers.
- **Common types:**
 - *Fabrication*: Inventing entities, dates, citations.
 - *Confabulation*: Combining unrelated facts into an incorrect claim.
 - *Assumptions*: Assuming with other entities and informations.
- **Mitigation approaches:** grounding with retrieved documents, model calibration, verification loops, and human-in-the-loop checks.

Examples

```
C:\Users\Lokesh>ollama run llama3.2
>>> what is the name of IIT in Andhra Pradesh?
The Indian Institute of Technology (IIT) in Andhra Pradesh is called IIT Hyderabad.

>>> what is the name of IIT in Telangana?
There is no IIT directly located in either Andhra Pradesh or Telangana, as both are part of the same state.

However, the Indian Institute of Technology (IIT) Kanpur has a satellite campus located at IIT Hyderabad, also known as IIT Hybrid Campus.
```

(a) Hallucination in LLama3.2 Model

More Examples



Prompt used: The Glenfinnan Viaduct is a historic railway bridge in Scotland, UK, that crosses over the west highland line between the towns of Mallaig and Fort William.

(b) Hallucination is Computer Vision SORA Model

```
C:\Users\laksh\llama\run_llama3.2
>>> tell me the recipe of wheatkheer
Wheat Kheer is a traditional Indian dessert made with wheat, milk, and sugar. Here's a simple recipe to make Wheat Kheer:

Ingredients:
• 2 cups wheat grains (raw or processed)
• 4 cups water
• 1 cup granulated sugar
• 1/2 teaspoon cardamom powder
• 1/4 teaspoon saffron threads (optional)
• 1 tablespoon ghee or unsalted butter
• Chopped nuts and dried fruit for garnish (optional)

Instructions:
1. Rinse the wheat grains in a fine mesh sieve until they are well-washed.
2. Soak the wheat grains in water for at least 4 hours or overnight. Drain and rinse them again.
3. In a large saucepan, combine the soaked wheat grains and 4 cups of fresh water. Bring to a boil over medium heat.
4. Reduce the heat to low and simmer for about 20-25 minutes, or until the wheat has broken down and the mixture has thickened into a creamy consistency.
5. Add the sugar, cardamom powder, and saffron threads (if using) to the saucepan. Mix well.
6. Cook for another 5-7 minutes, stirring occasionally, until the mixture is smooth and the sugar has dissolved.
7. Remove the saucepan from heat and stir in the ghee or butter until it's fully incorporated.
8. Let the kheer cool slightly before serving.
9. Garnish with chopped nuts and dried fruit (if desired) and serve warm or chilled.

Tips:
• You can use raw wheat grains or processed wheat flour for this recipe. Raw wheat grains will give a more authentic texture, while processed wheat flour will result in a smoother consistency.
• Adjust the amount of sugar to your taste. Some people prefer their kheer sweeter than others.
• You can also add other flavorings like cinnamon, ginger, or lemon zest to the kheer for added depth and complexity.
• Wheat Kheer is traditionally served on special occasions like weddings and festivals. It's a popular dessert in many parts of India, especially during the winter months.

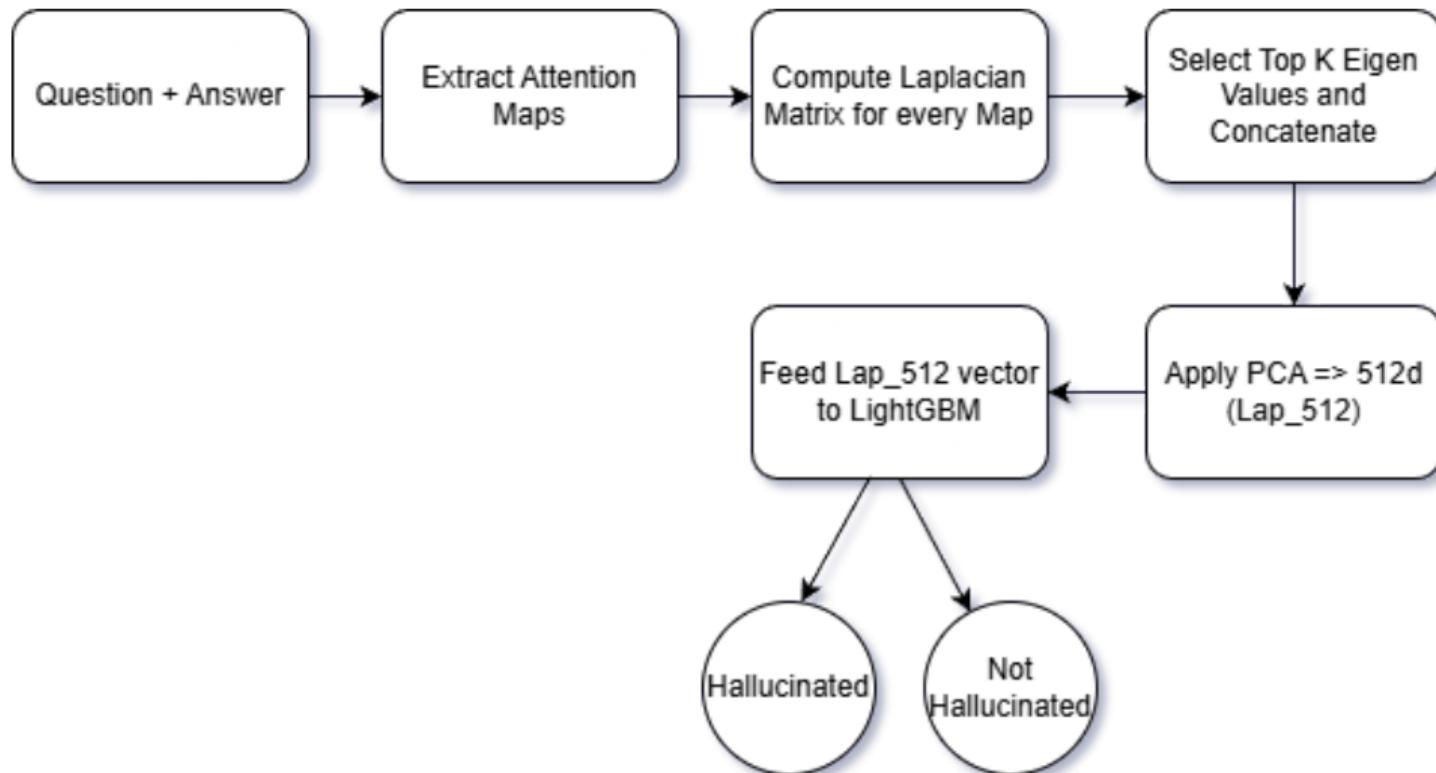
Enjoy your delicious homemade Wheat Kheer!
```

(c) Hallucination in LLama3.2 Model

Problem Statement

- Large Language Models (LLMs) often generate fluent but factually incorrect or fabricated content, known as hallucinations.
- Since Language Models works on the basis of predicting the most probable next word, if $P(\text{wrong word} \mid \text{context}) > P(\text{correct word} \mid \text{context})$, the model outputs the wrong word, prioritizing fluency and likelihood over factual accuracy.
- Such hallucinations reduce trust and limit adoption of LLMs in sensitive domains like healthcare, law, finance, and research.
- Detecting hallucinations is challenging because:
 - Ground-truth references are not always available in open-domain tasks.
 - Over-detection may reject valid answers, while under-detection risks factual errors.
 - Current solutions lack generalization across tasks and domains.
- **Research Problem:** To explore and design effective and generalizable methods and techniques to automatically detect hallucinations in LLM outputs, ensuring factual consistency and reliability?

LapEigvals: Existing Methodology



LapEigvals Cont..

- **Key idea:** Treat each attention map $A^{(l,h)} \in \mathbb{R}^{T \times T}$ (layer l , head h) as a weighted adjacency matrix of a directed graph, build a graph Laplacian and use its top- k eigenvalues as features for a hallucination probe (logistic regression).
- **Laplacian:**

$$L^{(l,h)} = D^{(l,h)} - A^{(l,h)}$$

where the diagonal degree entries are computed (paper) as

$$d_{ii}^{(l,h)} = \frac{\sum_u a_{ui}^{(l,h)}}{\sum_v \mathbf{1}\{a_{vi}^{(l,h)} \neq 0\}}.$$

- **Pipeline (short):**
 1. Extract attention $A^{(l,h)}$ for all layers/heads.
 2. Compute $L^{(l,h)}$, take eigenvalues (or diagonal if triangular), sort and take top- k per head.
 3. Concatenate features across heads/layers (optionally PCA) and train a probe.

LapEigvals Cont..

Toy sentence: “Blue car passes.”

Toy attention head with $T = 4$ tokens. Use a lower-triangular, row-stochastic attention matrix:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.2 & 0.8 & 0 & 0 \\ 0.1 & 0.2 & 0.7 & 0 \\ 0.05 & 0.15 & 0.2 & 0.6 \end{bmatrix} \quad (\text{each row sums to 1})$$

Step 1: column sums (sum over rows) and nonzero counts (per column):

$$\text{col_sums} = \sum_u a_{u,i} = [1.35, 1.15, 0.9, 0.6],$$

$$\text{nonzero_counts} = \#\{v : a_{v,i} \neq 0\} = [4, 3, 2, 1].$$

(These will be used to compute the diagonal degree entries.)

LapEigvals Cont..

Step 2: diagonal degree entries (paper formula)

$$d_{ii} = \frac{\text{col_sums}_i}{\text{nonzero_counts}_i} = \left[\frac{1.35}{4}, \frac{1.15}{3}, \frac{0.9}{2}, \frac{0.6}{1} \right] = [0.3375, 0.383333\dots, 0.45, 0.6].$$

So

$$D = \text{diag}(0.3375, 0.383333\dots, 0.45, 0.6).$$

Step 3: Laplacian $L = D - A$

$$L = \begin{bmatrix} 0.3375 & 0 & 0 & 0 \\ 0 & 0.38333 & 0 & 0 \\ 0 & 0 & 0.45 & 0 \\ 0 & 0 & 0 & 0.6 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.2 & 0.8 & 0 & 0 \\ 0.1 & 0.2 & 0.7 & 0 \\ 0.05 & 0.15 & 0.2 & 0.6 \end{bmatrix} = \begin{bmatrix} -0.6625 & 0 & 0 & 0 \\ -0.2 & -0.416666\dots & 0 & 0 \\ -0.1 & -0.2 & -0.25 & 0 \\ -0.05 & -0.15 & -0.2 & -0.2 \end{bmatrix}$$

Step 4: diagonal entries (Laplacian eigenvalues for this triangular case)

$$\text{diag}(L) = [-0.6625, -0.416666\dots, -0.25, 0.0].$$

LapEigvals Cont..

Step 5: sort (descending) and select top- k . Sorted (descending) diagonal entries:

$$[0.0, -0.25, -0.416666\dots, -0.6625].$$

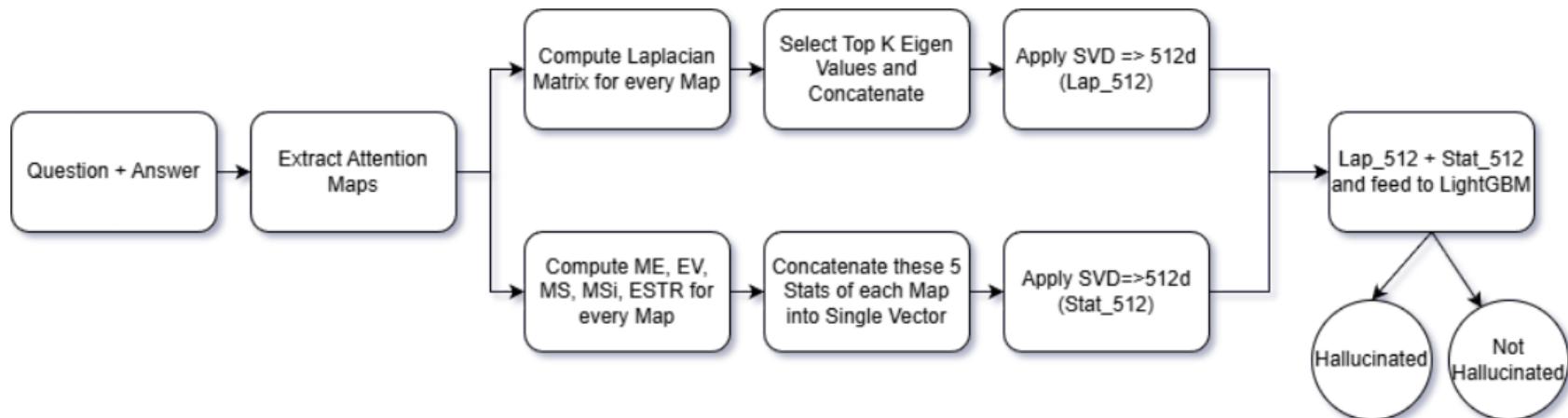
With $k = 2$:

$$\text{top-2} = [0.0, -0.25].$$

These selected top- k eigenvalues (from each head) are then concatenated across heads and layers, optionally reduced (PCA), and fed into a probe/classifier (e.g., logistic regression) to predict hallucination vs. truthful output.

Summary: In each layer/head we extract the attention matrix $A^{(l,h)}$, compute the graph Laplacian $L^{(l,h)} = D^{(l,h)} - A^{(l,h)}$, compute its eigenvalues and select the top- k per head, optionally apply dimensionality reduction (e.g., PCA), concatenate the resulting features across heads and layers, and feed the final vector to a binary classifier (e.g., logistic regression) to detect hallucinations.

Proposed Methodology



Method Cont..

Key idea: Extract compact per-head statistics from internal attention maps and combine them with spectral (Laplacian) features. These per-head statistics capture uncertainty (entropy) and structural similarity across layers; together they form a lightweight, interpretable feature set useful for hallucination detection.

Five per-head features (computed for each attention head):

1. Mean Entropy (ME)
2. Entropy Variance (EV)
3. Mean Similarity (MS)
4. Minimum Similarity (MSi)
5. Entropy-to-Similarity Ratio (ETSR)

Below we give precise formulas and short intuition for each.

Method Cont..

1. Mean Entropy (ME)

Per-row entropy:

$$H_i = - \sum_{j=1}^T A_{ij} \ln A_{ij} \quad (\text{use natural log})$$

Mean Entropy (average uncertainty across tokens):

$$\text{ME} = \mu_H = \frac{1}{T} \sum_{i=1}^T H_i.$$

Intuition: low ME \Rightarrow focused attention; high ME \Rightarrow diffuse/uncertain attention.

2. Entropy Variance (EV)

Variance of row entropies:

$$\text{EV} = \sigma_H^2 = \frac{1}{T} \sum_{i=1}^T (H_i - \mu_H)^2.$$

Intuition: measures consistency of attention-focus across tokens (large EV = uneven focus).

Method Cont..

Preprocessing for similarity features:

Flatten and ℓ_2 -normalize the attention matrix:

$$v = \text{vec}(A) \in \mathbb{R}^{T^2}, \quad \hat{v} = \frac{v}{\|v\|_2 + \varepsilon}.$$

3. Mean Similarity (MS)

For a head at layer ℓ with normalized vector $\hat{v}^{(\ell)}$, compute cosine similarity to every head p in layer $\ell - 1$ (previous layer):

$$s_p = \langle \hat{v}^{(\ell)}, \hat{v}_p^{(\ell-1)} \rangle.$$

Mean Similarity:

$$\text{MS} = \mu_s = \frac{1}{H} \sum_{p=1}^H s_p.$$

Intuition: how much this head's attention pattern resembles heads in previous layer (high = consistent pattern).

Method Cont..

4. Minimum Similarity (MSi)

Minimum cross-layer similarity:

$$\text{MSi} = S_{\min} = \min_p s_p.$$

Intuition: captures the weakest alignment to prior-layer heads (low MSi may indicate anomalous/noisy head).

5. Entropy-to-Similarity Ratio (ETSR)

Combine entropy and similarity into a single normalized ratio:

$$\text{ETSR} = \frac{\mu_s}{\mu_H + \varepsilon}.$$

Intuition: large ETSR indicates similarity dominates relative to entropy (structured attention), while small ETSR indicates entropy (diffuseness) dominates.

Summary

Dataset: HaluEval

HaluEval is a popular dataset widely used in the research topic of hallucinations. It contains around **10k samples**. The attributes are: **Question**, **Answer**, and **Hallucination Label**.

- Input: concatenate **Question + Answer**.
- Extract attention maps $A^{(l,h)}$ for all layers l and heads h .
- Two parallel feature paths:
 1. Spectral path: compute Laplacian for every attention map → take top- k eigenvalues per map → stack → apply SVD to reduce to 512-dim (Lap512).
 2. Statistical path: compute per-head statistics (Mean Entropy, Entropy Variance, Mean Similarity, Min Similarity, Entropy-to-Similarity Ratio) for every attention map → combine into a per-example matrix → apply SVD to reduce to 512-dim (Stat512).
- Concatenate Lap512 + Stat512 → final 1024-dim vector → feed to classifier (LightGBM).

Differences

Aspect	Existing Method	Proposed Method
Core pipeline	Single pipeline: only Laplacian eigenvalues are used.	Two parallel pipelines: (1) Laplacian eigenvalues, (2) statistical features.
Raw vector size	$L \times H \times K$ (spectral only).	Pipeline A (spectral): $L \times H \times K$. Pipeline B (statistical): $L \times H \times 5$.
Dimensionality reduction method	PCA to reduce spectral vector to 512 dims.	SVD on spectral vector → 512 dims; SVD on statistical matrix → 512 dims.
Final feature vector	Single 512-d vector (Lap512) fed to classifier.	Concatenate Lap512 + Stat512 → 1024 -d final vector.
Classifier input	512-d spectral embedding.	1024-d fused embedding (spectral + statistical).

Results

TinyLlama-1.3B (22 Layers, 32 Heads per Layer)

k Values	Accuracy		AUC-ROC	
	Existing	Proposed	Existing	Proposed
30	0.91	0.9465	0.958	0.9755
50	0.9185	0.9445	0.965	0.976
100	0.934	0.9435	0.958	0.959

GPT Neo-1.3B (24 Layers, 16 Heads per Layer)

k Values	Accuracy		AUC-ROC	
	Existing	Proposed	Existing	Proposed
30	0.8185	0.923	0.897	0.975
50	0.835	0.933	0.902	0.971
100	0.787	0.937	0.8544	0.964

References

-  Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention Is All You Need*. In *Advances in Neural Information Processing Systems (NeurIPS)*.
-  Binkowski, J., Janiak, D., Sawczyn, A., Gabrys, B., & Kajdanowicz, T. (2024). *Hallucination Detection in LLMs Using Spectral Features of Attention Maps*. Wroclaw University of Science and Technology; University of Technology Sydney.
-  Sriramanan, G., Saha, S., Bharti, S., Kattakinda, P., Sadasivan, V. S., & Feizi, S. (2024). *LLM-Check: Investigating Detection of Hallucinations in Large Language Models*. Department of Computer Science, University of Maryland, College Park, USA.
-  Su, W., Wang, C., Ai, Q., Hu, Y., Wu, Z., Zhou, Y., & Liu, Y. (2024). *Unsupervised Real-Time Hallucination Detection Based on the Internal States of Large Language Models*. Tsinghua University; Beijing Institute of Technology; Renmin University of China.

Future Work

- Extend the current system from **detection** to **mitigation** by generating answers for the HaluEval dataset and evaluating hallucination rate before and after mitigation.
- Apply **RAG-based regeneration** and **self-refinement** to reduce hallucinations in generated answers.
- Re-evaluate the model on HaluEval using the updated answers to measure how much the **hallucination rate decreases**.
- Integrate the mitigation step into the existing pipeline to build a full **detect → mitigate → re-detect** framework.

Thank You!