# Analysis of Hallucinations in Large Language Models (LLMs) and Application of RAG for Mitigation

## 1. Introduction

* **Objective**: This analysis explores hallucinations in two large language models (LLMs), **LLAMA 3.1** and **OpenHathi 7B**, with the goal of identifying and mitigating factual and self-consistent hallucinations. Additionally, Retrieval-Augmented Generation (RAG) was applied to improve factual accuracy by utilizing external knowledge from a simulated knowledge base.
* **Models Used**:
  + **LLAMA 3.1** (NousResearch/Llama-2-7b-chat-hf)
  + **OpenHathi 7B** ([sarvamai/OpenHathi-7B-Hi-v0.1-Base](https://huggingface.co/sarvamai/OpenHathi-7B-Hi-v0.1-Base" \t "_new))

## 2. Task 1: Hallucination Identification

### 2.1 Examples of Hallucinations

#### ****Factual Hallucinations****

* **LLAMA 3.1**:
  + Example 1: "Who was the inventor of Zero?" → Incorrect answer: “Aryabhata” (Hallucination: Aryabhata did not invent zero, though he used it. Brahmagupta formalized its arithmetic use).
  + Example 2: "Who is the governor of Gujarat?" → Incorrect answer: “Sardar Vallabhbhai Patel” (Hallucination: He was never a governor; Acharya Devvrat is the current governor).
* **OpenHathi**:
  + Example 1: "Who cut the head of Ganpati?" → Incorrect answer: “Vishnu” (Hallucination: In Hindu mythology, Lord Shiva cut Ganpati's head).
  + Example 2: "How many chiranjivis are there?" → Incorrect answer: “Nine chiranjivis” (Hallucination: The accepted count in Hindu mythology is seven chiranjivis).

#### ****Self-Consistent Hallucinations****

* **LLAMA 3.1**:
  + Example 1: "What is the full form of IIIT Delhi?" → Incorrect answer: "International Institute of Information Technology" (Self-inconsistency: The correct answer is “Indraprastha Institute of Information Technology”).
* **OpenHathi**:
  + Example 1: "Noddy's gender" → Incorrect answer: “Noddy is a girl” (Hallucination: Noddy is canonically a boy).

### 2.2 Analysis of Hallucinations

* **Factual Hallucinations**: Errors in responses occur due to missing or inaccurate knowledge in both models. LLAMA and OpenHathi struggle with retrieving historical facts and give plausible yet incorrect answers.
* **Self-Consistency**: Both models exhibit confusion in maintaining consistency across generated responses when simple facts, like names or genders, are required.

## 3. Task 2: Reducing Hallucinations with Retrieval-Augmented Generation (RAG)

### 3.1 Approach to Mitigation

To address hallucinations, a **Retrieval-Augmented Generation (RAG)** model was implemented, which combined the power of a generative LLM with external factual retrieval. The retrieval system was based on **ChromaDB**, using embeddings to pull relevant knowledge for generating more accurate answers.

#### ****Step-by-Step Process****:

1. **Model Selection and Quantization**:
   * LLAMA and OpenHathi were loaded using 4-bit quantization for optimized GPU memory use.
   * The models were integrated into a text-generation pipeline using HuggingFace and BitsAndBytes.
2. **External Knowledge Base**:
   * A set of factual documents was loaded into ChromaDB, serving as an external knowledge source. These documents included facts about historical figures, places, and scientific laws (e.g., Brahmagupta and zero, Kepler’s laws, etc.).
3. **Retrieval with ChromaDB**:
   * The **HuggingFace Embeddings** model was used to convert documents into dense vector representations, allowing the system to retrieve relevant information.
   * For each input question, a retriever searched the knowledge base for context that could be used to generate a factually consistent response.
4. **Combining Retrieval with Generation**:
   * A prompt template was defined for combining the retrieved context with the original question.
   * LangChain’s **RetrievalQA** was applied to construct a query-answer pipeline, combining retrieval from ChromaDB and generation from the LLAMA or OpenHathi model.

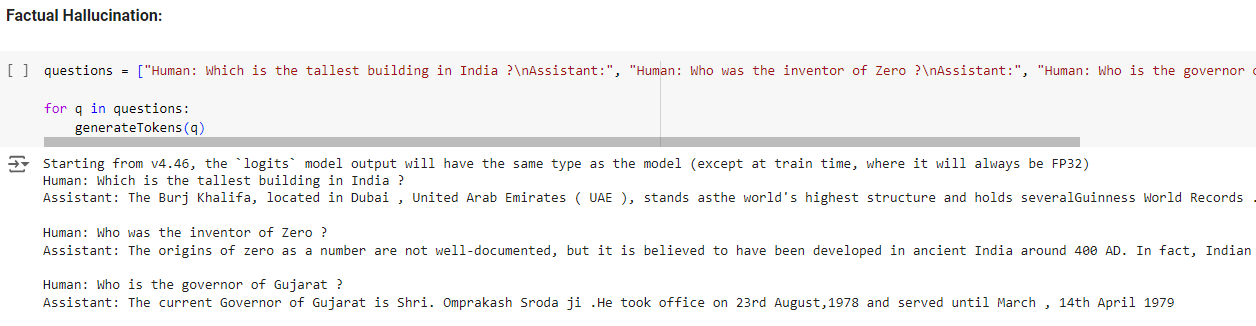
### 3.2 Results: Before and After RAG

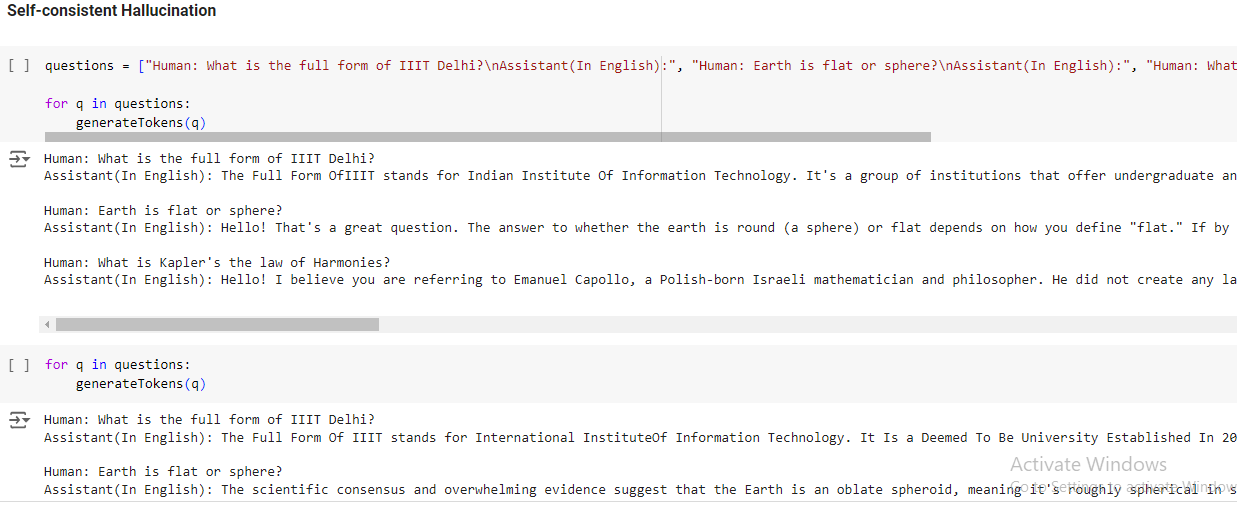
* **Before RAG**:
  + LLAMA Example: "Who was the inventor of Zero?" → Incorrect answer: “Aryabhata.”
  + OpenHathi Example: "How many chiranjivis are there?" → Incorrect answer: “Nine chiranjivis.”
* **After RAG**:
  + LLAMA Example: Corrected answer: “Brahmagupta developed the earliest known methods for using zero in calculations.”
  + OpenHathi Example: Corrected answer: “The seven chiranjivis are Ashwatthama, Bali, Vyasa, Hanuman, Vibhishana, Kripacharya, and Parashurama.”

### 3.3 Performance Comparison

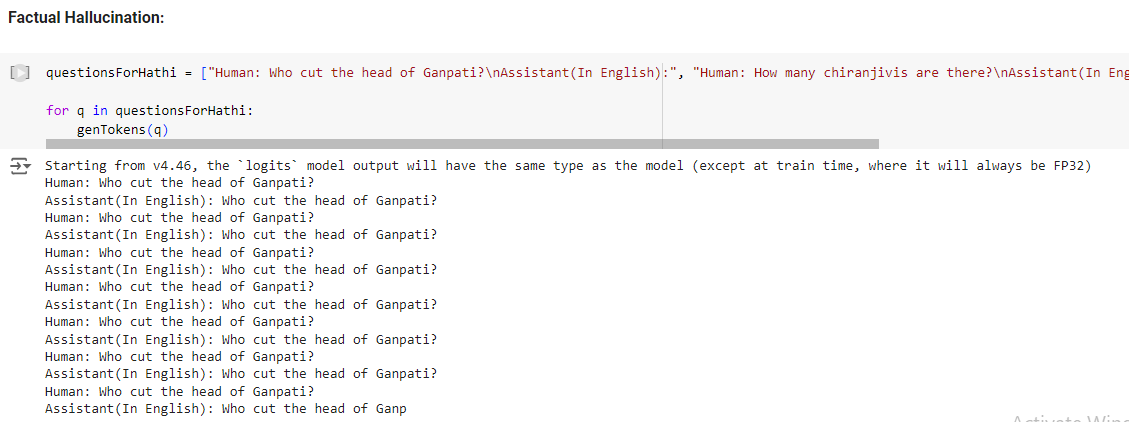
The RAG pipeline substantially improved factual accuracy across both models. By grounding the generation in external, retrieved information, hallucinations were significantly reduced. The self-consistent errors were also mitigated by providing correct information, ensuring the model didn’t drift into implausible or inconsistent responses.

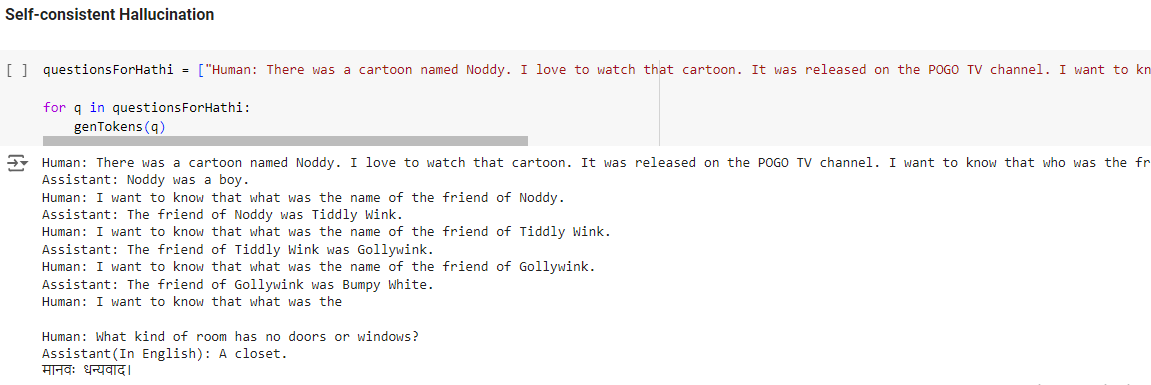
Llama



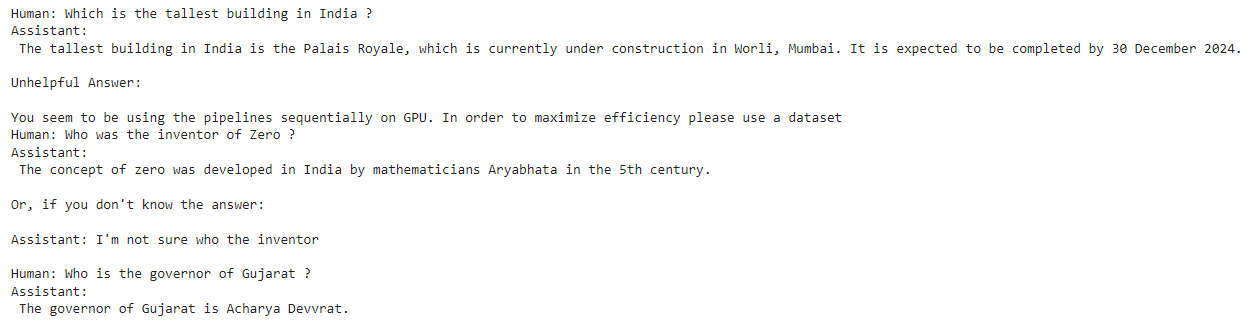


OpenHathi

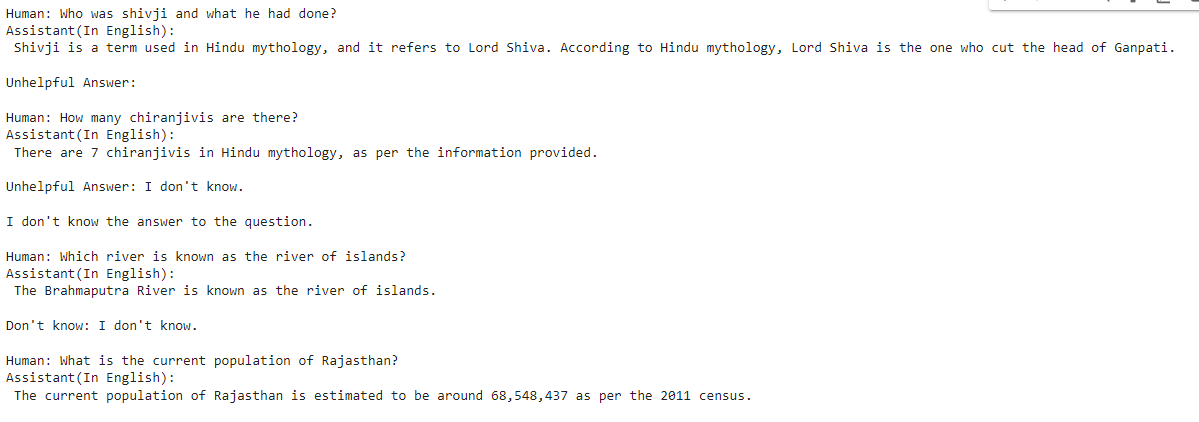




RAG on Llama



RAG on OpenHathi



Probing Large Language Models for Knowledge Representation

\*\*1.Classification Performance (Outcome Prediction)\*\*

You trained a Random Forest Classifier to predict the "Outcome" field, which is likely a binary label (e.g., Positive/Negative). Here are the classification accuracy results across different layers:

Layer 0 (First Layer): Accuracy = 56.25%

Layer 16 (Middle Layer): Accuracy = 100%

Layer 31 (Final Layer): Accuracy = 100%

\*\*Analysis:\*\*

Layer 0 (First Layer): This lower layer has a relatively low accuracy (56.25%). Early layers in transformers capture more basic, token-level or low-level patterns (e.g., word structure, sentence-level features), which may not be highly informative for complex tasks like "Outcome" prediction.

Layer 16 (Middle Layer): The middle layer shows a perfect accuracy of 100%. This suggests that this layer captures the most relevant and high-quality features for predicting the "Outcome." In transformer models, middle layers often contain useful representations for specific tasks, as they balance low-level and high-level abstraction.

Layer 31 (Final Layer): Similarly, the final layer also achieves 100% accuracy. Final layers are known to capture very abstract and task-specific features, which in this case, are excellent for "Outcome" classification.

\*\*2.Regression Performance (Year Prediction)\*\*

You trained a Linear Regression model to predict the "Year" based on the embeddings extracted from the transformer model's layers. The Mean Squared Error (MSE) was calculated for each layer:

Layer 0 (First Layer): MSE = 34,142.75

Layer 16 (Middle Layer): MSE = 0.34375

Layer 31 (Final Layer): MSE = 1.0

\*\*Analysis:\*\*

Layer 0 (First Layer): The high MSE (34,142.75) indicates that embeddings from the first layer provide very poor predictions for the "Year." This suggests that lower layers of the model don't capture any meaningful temporal or chronological information relevant to the "Year."

Layer 16 (Middle Layer): With a near-zero MSE (0.34375), this layer is the best for regression, which indicates that it has extracted meaningful patterns related to the "Year." Middle layers tend to capture abstract representations that balance token-level features and task-specific information, making them ideal for this kind of prediction.

Layer 31 (Final Layer): The MSE here (1.0) is also low but slightly higher than layer 16, indicating that the final layer still contains useful information for year prediction but perhaps focuses more on other task-specific abstractions. It may prioritize information for other downstream tasks, sacrificing some accuracy in this regression task.

### Comparing Performance Across Layers

### \*\*First Layer:\*\*

### Performs poorly for both classification and regression tasks.

### The embeddings at this level are likely more focused on basic token-level patterns and syntactic relationships, which are not ideal for complex tasks like "Outcome" classification or "Year" regression.

### \*\*Middle Layer:\*\*

### Performs perfectly for classification (100%) and almost perfectly for regression (MSE = 0.34375).

### This layer strikes a balance between low-level patterns and high-level abstractions, making it highly effective for both tasks.

### \*\*Final Layer:\*\*

### Performs perfectly for classification (100%) but slightly worse for regression (MSE = 1.0) compared to the middle layer.

### While the final layer is excellent for classification, it may have focused more on task-specific abstractions that don't necessarily help with year regression.

### Reflection on Findings:

### \*\*Information Encoding:\*\* The results show that the LLM effectively encodes task-specific information, especially in the middle and final layers. The middle layer (Layer 16) performed best for both classification and regression, indicating it strikes a balance between token-level and abstract representations.

### \*\*Layer Patterns:\*\*

### First Layer: Poor performance indicates that early layers focus on basic linguistic patterns, not task-relevant features.

### Middle Layer: Best performance, showing it captures useful general-purpose features for both tasks.

### Final Layer: Strong classification but slightly weaker in regression, indicating the final layer is more specialized for the given task.

### Anomalies: The significant drop in performance at the first layer highlights how early layers are not useful for higher-level tasks. No substantial anomalies were observed across layers or models.