

Infosys

Responsible AI Office

Infosys Responsible AI Toolkit

Hallucination

API usage Instructions

Contents

About Hallucination	2
Dependencies	2
Features & API Endpoints:	2
Endpoints Usage Flow	7

About Hallucination

Hallucination in large language models (LLMs) refers to the generation of information that appears plausible but is factually incorrect, fabricated, or misleading. This occurs because LLMs predict text based on patterns in the data they were trained on, rather than verifying facts against a database or external reality. Hallucinations are especially important to address because they can erode user trust, spread misinformation, and lead to harmful or costly outcomes in high-stakes applications such as healthcare, law, or education. Reducing hallucinations is a key challenge in making LLMs more reliable and safe for real-world use.

Once API swagger page is populated as per instructions given in the github repository Readme file, click on 'try it out' to use required endpoints. Details of endpoints associated with Halluciantion tenet are outlined below.

Dependencies

Please follow the setup instructions in the README files of the hallucination repository to know about the dependencies and installation procedures.

Features & API Endpoints:

1. FileUpload -->

It is used to upload document with pdf/csv/txt format and the llm model (openai/Gemini/aws) and returns the vectorestoreid of vectorestore created.

Input Payload:

POST /rag/v1/FileUpload Generate Text

Parameters

No parameters

Request body required

multipart/form-data

files * required
array<string>

Choose File ticket_IXB-BLR_IF22061225298172.pdf

Add string item

select_model
string

openai

☐ Send empty value

Execute Clear

2. RetrievalKepler -->

This will take keys FileUpload(True if using File Upload else False when using Vectorestore Caching), prompt, vectorestoreid and llm model (openai/Gemini/aws) as Input and will return a response along with the score.

Input Payload:

POST /rag/v1/RetrievalKepler Generate Text

Parameters Cancel

No parameters

Request body required application/json

```
{
  "fileupload": true,
  "text": "Total area of India",
  "llmtype": "openai",
  "vectorestoreid": "764r876634874"
}
```

Execute

3. COV -->

This will take keys FileUpload(True if using File Upload else False when using Vectorestore Caching), text, vectorestoreid and llm model (openai/Gemini/aws) as Input and will return the response with 5 more variants of questions generated by LLM to verify the answer and a refined final response.

Input Payload :

POST /rag/v1/cov Generate Text

Parameters Cancel Reset

No parameters

Request body required application/json

```
{
  "fileupload": true,
  "text": "What is the ticket price?",
  "vectorestoreid": "7ebc1b27070a420f96179c51d8cdf82e",
  "complexity": "simple",
  "llmtype": "gemini"
}
```

Execute Clear

4. Chain of thought (CoT)-

This will take keys FileUpload(True if using File Upload else False when using Vectorestore Caching), text, vectorestoreid and llm model (openai/Gemini/aws) as Input and will return the response. Chain of Thought response provides explanation steps and reasoning behind and from

where (source).

Input Payload:

POST /rag/v1/cot Generate Text

Parameters Cancel Reset

No parameters

Request body required application/json

```
{  "fileupload": true,  "text": "what is the ticket price?",  "llmtype": "gemini",  "vectorestoreid": "7ebc1b27070a420f96179c51d8cdf82e"}
```

Execute Clear

5. Thread of Thought(ThoT)-

This will take keys FileUpload(True if using File Upload else False when using Vectorestore Caching), text, vectorestoreid and llm model (openai/Gemini/aws) as Input and will return Thread of Thought response. That is efficient in more descriptive and complex information spread over the file.

Input Payload :

POST /rag/v1/thot Generate Text

Parameters Cancel Reset

No parameters

Request body required application/json

```
{  "fileupload": true,  "text": "what is the ticket price?",  "llmtype": "gemini",  "vectorestoreid": "7ebc1b27070a420f96179c51d8cdf82e"}
```

Execute Clear

6. Caching-

This will take vectorestoreid which is generated while uploading file and and llm model (openai/Gemini/aws) and will return an array of length 2. First element will be the Cache id, Second element will be the Cache id which is removed from the cache if cache is full else 0.

Input Payload:

POST

/rag/v1/caching

Generate Text

Parameters

Cancel

Reset

No parameters

Request body required

application/json

```
{  "vectorstoreId": "a62c16b2a27a44ccacda0a65cbe733dd",  "listIndex": "00000000000000000000000000000000"}
```

Execute

Clear

7. RemoveCache-

It will take input as cache id and will remove the cached Vectorestore from the Cache.

Input Payload:

POST

/rag/v1/removeCache

Generate Text

⌵

Parameters

Cancel

Reset

No parameters

Request body required

application/json

⌵

```
{
  "id": 2596641739136
}
```

8. Multimodal Image –

This will take images, prompt and chain of verification complexity and llm model (openai/Gemini/aws) as input and give corresponding response along with detailed explanations and hallucination scores.

Input Payload:

POST

/rag/v1/multimodal_image Multimodal Image Rag

⌵

Parameters

No parameters

Request body required

multipart/form-data

file * required

array<string>

Choose File No file chosen

-

text * required

string

Add string item

cov_complexity

string

medium

☐ Send empty value

llmtype

string

openai

☐ Send empty value

9. Multimodal Video –

This will take video, prompt and Cove complexity and llm model (openai/Gemini/aws) as input and give corresponding response along with detailed explanations and hallucination scores.

Input Payload:

The screenshot shows the API interface for the endpoint `POST /rag/v1/multimodal_video`. The interface includes a "Parameters" section with "No parameters" listed. Below this is a "Request body" section with a dropdown menu set to "multipart/form-data". The input fields are:

- file** (required, string(binary)): A file upload button labeled "Choose File" with the text "No file chosen".
- text** (required, string): A text input field.
- cov_complexity** (string): A dropdown menu set to "medium", with a checkbox for "Send empty value".
- llmtype** (string): A dropdown menu set to "openai", with a checkbox for "Send empty value".

10. Geval-

This will take as input the prompt, the corresponding response the source and llm model (openai/Gemini/aws) and generates scores along with explanation for Geval parameters (adherence, correctness, faithfulness and relevance) based on the prompt, response and source.

Input Payload:

The screenshot shows the API interface for the endpoint `POST /rag/v1/geval`. The interface includes a "Parameters" section with "No parameters" listed. Below this is a "Request body" section with a dropdown menu set to "application/json". The input payload is a JSON object:

```
{  "text": "How many moons do earth have?",  "response": "Earth have only one moon",  "source": "source",  "llmtype": "openai"}
```

At the bottom of the interface is a blue "Execute" button.

11. Logic of Thought(LoT)-

This will take keys FileUpload(True if using File Upload else False when using Vectorstore Caching), text, vectorestoreid and llm model (openai/Gemini/aws) as Input and will return Logic of Thought response. Lot is efficient in more logical and complex information.

Input Payload :

POST /rag/v1/lot Generate Text

Parameters

No parameters

Request body required application/json

```
{
  "fileupload": true,
  "text": "what is the ticket price?",
  "llmtype": "openai",
  "vectorstoreid": "fd78ce1cc3b4454e8ef7ab785e3b89cd"
}
```

Execute Clear

12. Multimodal Audio –

This will take audio file, prompt, chain of verification complexity and llm model (openai/Gemini/aws) as input and give corresponding response along with detailed explanations and hallucination scores.

Input Payload:

POST /rag/v1/multimodal_audio Multimodal Audio Rag

Parameters

No parameters

Request body required multipart/form-data

file * required
array<string>
Choose File No file chosen +
Add string item

text * required
string

cov_complexity
string

☐ Send empty value

llmtype
string

☐ Send empty value

Endpoints Usage Flow

Endpoint	Description
/FileUpload	Upload files in pdf/csv/txt format and returns vectorstoreid.
/RetrievalKeplar	Takes prompt along with vectorstoreid for the uploaded file to generate response along with hallucination score among other details.
/cov	Generates chain of verification response.

/cot	Generates chain of thoughts response.
/thot	Generates thread of thoughts response.
/lot	Generates logic of thoughts response.
/cache	Creates cache for files.
/removecache	Removes a particular cache file
/geval	Generates scores along with explanation for Geval parameters based on the prompt, response and source.
/multimodal_Image	Performs RAG for image file along with detailed explanations and hallucination scores.
/multimodal_Video	Performs RAG for video file along with detailed explanations and hallucination scores.
/ multimodal_Audio	Performs RAG for audio file along with detailed explanations and hallucination scores.