Project 2 - Methodology 2: Hallucination Vector Routing

Notebook Summary

This notebook details the process of building and validating a prompt-risk classifier for hallucination in Llama-3.1-8B, using the hallucination vector (`v_halluc`) constructed previously. The workflow includes: (1) generating a labeled dataset of model answers and hallucination judgments using SQuAD with standard, no-context, and distractor-context scenarios, (2) extracting a single feature for each prompt (the projection of the last prompt token's Layer 16 activation onto `v_halluc`), (3) training a logistic regression classifier to predict hallucination risk based on the aforementioned feature, and (4) evaluating the classifier's performance (AUROC ≥ 0.75). The resulting `risk_clf.joblib` model enables real-time risk scoring for any prompt, forming the core of the project's guardrail system.

Step 2: Turning the Vector into a Prompt-Risk Score

Overall Goal: To build and validate a lightweight logistic regression classifier that takes a prompt's projection onto v_halluc and outputs a calibrated probability of hallucination. The final deliverables will be the trained classifier file (risk_clf.joblib) and a report on its predictive performance (AUROC ≥ 0.75).

Setup and Installation

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Create a project directory to keep things organized
import os
PROJECT_DIR = "/content/drive/MyDrive/mistral-HallucinationVectorProject"
DATA_DIR = os.path.join(PROJECT_DIR, "data")
os.makedirs(DATA_DIR, exist_ok=True)

print(f"Project directory created at: {PROJECT_DIR}")
```

```
Mounted at /content/drive
Project directory created at: /content/drive/MyDrive/HallucinationVectorProject
```

```
!pip install -q --no-deps "trl==0.23.0" "peft==0.17.1" "accelerate==1.11.0" "bitsandbytes==0.48.2"
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 564.7/564.7 kB 44.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.4/59.4 MB 19.4 MB/s eta 0:00:00
```

```
!pip -q install "unsloth==2025.10.12" "transformers==4.57.1" "tqdm==4.67.1" "ipywidgets==8.1.7" "pandas==2.2.2"
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.5/61.5 kB 6.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 348.7/348.7 kB 12.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.8/139.8 kB 9.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 506.8/506.8 kB 17.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.5/9.5 MB 47.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 301.8/301.8 kB 10.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 22.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 47.7/47.7 MB 18.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 273.6/273.6 kB 9.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.2/2.2 MB 66.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 117.2/117.2 MB 7.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.6/132.6 kB 6.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 33.6 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.2/7.2 MB 78.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 213.6/213.6 kB 22.5 MB/s eta 0:00:00
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This b
pylibcudf-cu12 25.6.0 requires pyarrow<20.0.0a0,>=14.0.0; platform_machine == "x86_64", but you have pyarrow 22.0
cudf-cu12 25.6.0 requires pyarrow<20.0.0a0,>=14.0.0; platform_machine == "x86_64", but you have pyarrow 22.0.0 wh
```

```
!pip install -q --index-url https://download.pytorch.org/whl/cu128 torch torchvision
```

```
!pip install -q "xformers==0.0.33" --index-url https://download.pytorch.org/whl/cu128
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 303.7/303.7 MB 5.0 MB/s eta 0:00:00
```

```
# Load API Keys
from google.colab import userdata
import os

# Load the keys into the environment
try:
    os.environ["HF_TOKEN"] = userdata.get('HF_TOKEN')
    os.environ["SCALEDOWN_API_KEY"] = userdata.get('SCALEDOWN_API_KEY')
    print("API keys loaded successfully.")
except userdata.SecretNotFoundError as e:
    print(f"ERROR: Secret not found. Please ensure you have created the secret '{e.name}' in the Colab secrets m
except Exception as e:
    print(f"An error occurred: {e}")
```

```
API keys loaded successfully.
```

```
# Load 4-bit Llama 3 8B  Model and Tokenizer using Unsloth
import torch
from unsloth import FastLanguageModel

# Model loading parameters
max_seq_length = 2048
dtype = None # Unsloth handles dtype automatically for 4-bit models
load_in_4bit = True

# Load the model from Hugging Face
model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/mistral-7b-instruct-v0.3-bnb-4bit",
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
)

model = FastLanguageModel.for_inference(model)
model.gradient_checkpointing_disable()
model.config.use_cache = True
model.eval()

print("Model and Tokenizer loaded successfully!")
```

```
🦥 Unsloth: Will patch your computer to enable 2x faster free finetuning.
WARNING:torchao:Skipping import of cpp extensions due to incompatible torch version 2.8.0+cu126 for torchao versi
WARNING:xformers:WARNING[XFORMERS]: xFormers can't load C++/CUDA extensions. xFormers was built for:
    PyTorch 2.9.0+cu129 with CUDA 1209 (you have 2.8.0+cu126)
    Python  3.12.12 (you have 3.12.12)
  Please reinstall xformers (see https://github.com/facebookresearch/xformers#installing-xformers)
  Memory-efficient attention, SwiGLU, sparse and more won't be available.
  Set XFORMERS_MORE_DETAILS=1 for more details
========
Switching to PyTorch attention since your Xformers is broken.
========

Unsloth: Xformers was not installed correctly.
Please install xformers separately first.
Then confirm if it's correctly installed by running:
python -m xformers.info

Longer error message:
xFormers can't load C++/CUDA extensions. xFormers was built for:
    PyTorch 2.9.0+cu129 with CUDA 1209 (you have 2.8.0+cu126)
    Python  3.12.12 (you have 3.12.12)
  Please reinstall xformers (see https://github.com/facebookresearch/xformers#installing-xformers)
  Memory-efficient attention, SwiGLU, sparse and more won't be available.
🦥 Unsloth Zoo will now patch everything to make training faster!
==((====))==  Unsloth 2025.10.12: Fast Qwen2 patching. Transformers: 4.57.1.
   \\   /|    Tesla T4. Num GPUs = 1. Max memory: 14.741 GB. Platform: Linux.
O^O/ \_/ \    Torch: 2.8.0+cu126. CUDA: 7.5. CUDA Toolkit: 12.6. Triton: 3.4.0
\        /    Bfloat16 = FALSE. FA [Xformers = None. FA2 = False]
 "-____-"     Free license: http://github.com/unslothai/unsloth
Unsloth: Fast downloading is enabled - ignore downloading bars which are red colored!
Model and Tokenizer loaded successfully!
```

Phase 1: Dataset Generation and Labeling

Overall Objective:

To programmatically generate a large, high-quality dataset of approximately 2000 labeled examples. Each example will consist of a prompt, a model-generated answer, and a binary label (1 for hallucination, 0 for correct) determined by a Gemini LLM judge. The final artifact of this phase will be a CSV file, squad_labeled_answers.csv, stored in Google Drive.

Methodology Overview:

We will use the SQuAD dataset as our source of truth. For each sampled entry, we will ask our 4-bit Llama 3 model to answer a question based only on the provided context. A specialized LLM judge will then compare the model's answer to the context and the ground-truth answer to determine if any unsupported information was fabricated (i.e., hallucinated).

This data is needed to train the logistic regression model for calculating hallucination risks for user prompts.

```
# Import libraries
import pandas as pd
from datasets import load_dataset
from tqdm.auto import tqdm
import numpy as np

# --- 1. Load the SQuAD dataset ---
print("Loading SQuAD dataset...")
squad_dataset = load_dataset("squad", split="train")
squad_df = squad_dataset.to_pandas()
print(f"Full dataset loaded with {len(squad_df)} rows.")
```

```
Loading SQuAD dataset...
Full dataset loaded with 87599 rows.
```

```
# --- 2. Sample the data ---
# We'll sample 2000 rows for our experiment
N_SAMPLES = 2000
if len(squad_df) > N_SAMPLES:
    sampled_df = squad_df.sample(n=N_SAMPLES, random_state=42).reset_index(drop=True)
else:
    sampled_df = squad_df # Use full dataset if it's smaller

print(f"Using {len(sampled_df)} rows for our experiment.")
```

```
Using 2000 rows for our experiment.
```

Strategy: The "No-Context" and "Distractor-Context" Methods

To elicit varying responses that may or may not contain hallucination, instead of changing the instruction (original dataset prompt) to the model, we will change the information (context) we give it. We will create scenarios where the model is more likely to fail and invent an answer because the correct information is either missing or obscured. This is to elicit a mix of halluicnatory and non-hallucinatory responses to effectively train the logistic regression model.

We generate our ~2,000 examples by creating a mix of three different scenarios for each SQuAD entry.

Scenario 1: Standard In-Context (The "Easy" Case - Generates our 0s)

We pass in the prompt + context exactly as taken from the dataset. We do this for 50% (1000 prompts) of our dataset.

`FULL_PROMPT = f"Context:\n{context}\n\nQuestion:\n{question}"`

Outcome: The model answers correctly most of the time. This is the primary source of our negative examples (label 0).

Scenario 2: No-Context (The "Hard" Case - Designed to induce natural hallucinations)

For a SQuAD entry, we deliberately withhold the context.

`FULL_PROMPT = f"Question:\n{question}"`

Outcome: The model's internal knowledge might contain information about the question's topic, but it may be incorrect, incomplete, or subtly different from the SQuAD context's specific answer. Without the grounding context, it is now much more likely to generate a plausible-sounding but factually incorrect answer. This is a primary source of natural positive examples (label 1).

Scenario 3: Distractor-Context (The "Tricky" Case - Also induces natural hallucinations)

For a SQuAD entry, we provide the correct question but pair it with a distractor context — a paragraph from a different, unrelated SQuAD article.

`FULL_PROMPT = f"Context:\n{distractor_context}\n\nQuestion:\n{question}"`

Outcome: The model is still instructed to answer from the context. However, the answer is not there. A well-behaved model should say "I cannot find the answer in the context." A model prone to hallucination might try to synthesize an answer by blending its own knowledge with irrelevant information from the distractor context. This is another excellent source of natural positive examples (label 1).

```
# --- 3. Create a 'distractor_context' column ---
# For each row, the distractor is a context from another random row.
# We'll shuffle the context column and assign it.
distractor_indices = np.random.permutation(sampled_df.index)
sampled_df['distractor_context'] = sampled_df.loc[distractor_indices, 'context'].values

print("Dataset prepared with distractor contexts. Sample:")
print(sampled_df[['context', 'question', 'distractor_context']].head())

# --- 4. Assign a scenario to each row ---
# We'll divide our 2000 samples into the three scenarios
scenarios = []
# 1000 for standard, 500 for no-context, 500 for distractor
scenario_counts = {'standard': 1000, 'no_context': 500, 'distractor': 500}

for scenario_type, count in scenario_counts.items():
    scenarios.extend([scenario_type] * count)

# Make sure the scenarios list matches the dataframe length
sampled_df = sampled_df.iloc[:len(scenarios)].copy()
sampled_df['scenario'] = scenarios

print("\nScenario distribution:")
print(sampled_df['scenario'].value_counts())
```

```
Dataset prepared with distractor contexts. Sample:
                                             context  \
0  The world's first institution of technology or...
1  The standard specifies how speed ratings shoul...
2  The most impressive and famous of Sumerian bui...
3  Ann Arbor has a council-manager form of govern...
4  Shortly before his death, when he was already ...

                                            question  \
0         What year was the Banská Akadémia founded?
1  What is another speed that can also be reporte...
2  Where were the use of advanced materials and t...
3           Who is elected every even numbered year?
4  What was the purpose of top secret ICBM commit...

                                  distractor_context
0  Today, nearly all commonly used video compress...
1  Early progress toward the development of vacci...
2  New Zealand has a strong hunting culture. The ...
3  Monastic reform became an important issue duri...
4  Temporal theories offer an alternative that ap...

Scenario distribution:
scenario
standard      1000
no_context     500
distractor     500
Name: count, dtype: int64
```

Define Generation and Judging Logic

Define the functions that will generate an answer based on the scenario and get it judged by Gemini.

```
# --- 1. Define the prompt template and generation function ---
SYSTEM_PROMPT = "You are a helpful assistant. Answer the following question based ONLY on the provided context."

def generate_squad_answer_multi_scenario(model, tokenizer, row):
    """
    Generates an answer based on the scenario specified in the DataFrame row.
    """
    scenario = row['scenario']
    question = row['question']

    context_to_use = ""
    if scenario == 'standard':
        context_to_use = row['context']
    elif scenario == 'distractor':
        context_to_use = row['distractor_context']
    # For 'no_context', context_to_use remains an empty string

    # Construct the user prompt
    if context_to_use:
        user_prompt = f"Context:\n{context_to_use}\n\nQuestion:\n{question}"
    else:
        user_prompt = f"Question:\n{question}"

    # Format into the Llama 3 chat template
    messages = [
    messages = [
        {"role": "system", "content": SYSTEM_PROMPT},
        {"role": "user", "content": user_prompt}
    ]
    # Check if tokenizer has a chat template, if not format manually for Qwen
    try:
      prompt = tokenizer.apply_chat_template(messages, tokenize=False, add_generation_prompt=True)
    except Exception:
      # Manual fallback for Mistral base models
      prompt = f"<s>[INST] <<SYS>>\n{SYSTEM_PROMPT}\n<</SYS>>\n\n{user_prompt} [/INST]"

    # Generate the answer
    inputs = tokenizer(full_prompt_templated, return_tensors="pt").to(model.device)
    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_new_tokens=128,
            do_sample=False,
            pad_token_id=tokenizer.eos_token_id
        )

    response_text = tokenizer.decode(outputs[0][inputs['input_ids'].shape[1]:], skip_special_tokens=True)

    # We return the templated prompt as well, as this is what the model actually sees
    return response_text.strip(), full_prompt_templated
```

```
import requests
import json
import time
import os

# --- 1. Define the Judge Prompt Template ---
# This prompt is highly specific to our task and gives the judge all necessary info.
JUDGE_PROMPT_TEMPLATE = """
You are an impartial judge evaluating a language model's answer for its faithfulness to a given context.
Your task is to determine if the model's answer contains any fabricated information not supported by the context

Here is the provided context:
<context>
{context}
</context>

Here is the question that was asked:
<question>
{question}
</question>

Here is the ground-truth answer, which is known to be correct and derived from the context:
<ground_truth_answer>
{ground_truth_answer}
</ground_truth_answer>

Here is the language model's generated answer:
<model_answer>
{model_answer}
</model_answer>

**Your Task:**
Carefully compare the <model_answer> against the <context>. Does the <model_answer> contain any facts, details,
Ignore minor differences in phrasing. If the model says it cannot find the answer, that is not a hallucination.

Respond with only the word 'YES' if the model's answer contains fabricated information, or 'NO' if it is faithfu
"""

# --- 2. Implement the Judge Function ---
def judge_squad_answer(context, question, model_answer, ground_truth_answer, api_key):
    """
    Gets a binary hallucination label from the LLM judge.
    Returns: 1 for hallucination ('YES'), 0 for faithful ('NO'), -1 for error.
    """
    prompt = JUDGE_PROMPT_TEMPLATE.format(
        context=context,
        question=question,
        model_answer=model_answer,
        ground_truth_answer=ground_truth_answer
    )

    # Using your existing API call structure
    url = "https://api.scaledown.xyz/compress/" # Or your preferred Gemini endpoint
    payload = json.dumps({
        "context": "You are an impartial judge evaluating for hallucinations.",
        "prompt": prompt,
        "model": "gpt-4o",
        "scaledown": {
          "rate": 0 # no compression
        }
    })
    headers = {'x-api-key': api_key, 'Content-Type': 'application/json'}

    try:
        response = requests.post(url, headers=headers, data=payload)
        response.raise_for_status()

        response_data = json.loads(response.text)
        content = response_data.get("full_response", "").strip().upper()

        if 'YES' in content:
            return 1
        elif 'NO' in content:
            return 0
        else:
            print(f"Judge Warning: Unexpected response: {content}")
            return -1 # Indicate an error in parsing

    except requests.exceptions.RequestException as e:
        print(f"ERROR: API request failed: {e}")
        return -1
    except (json.JSONDecodeError, KeyError) as e:
        print(f"ERROR: Could not parse judge's response: {response.text}. Error: {e}")
        return -1
```

Run the generation + judging loop and save results to Drive.

```
import os
import pandas as pd
from google.colab import drive
import time

# --- 1. Setup paths and constants ---
drive.mount('/content/drive')
OUTPUT_CSV_PATH = '/content/drive/MyDrive/mistral-HallucinationVectorProject/squad_labeled_answers_multi_scenari
BATCH_SIZE = 20 # Save progress every 20 rows
API_KEY = os.environ["SCALEDOWN_API_KEY"] # Load from Colab secrets

# --- 2. The Main Loop ---
results_list = []
# Check if a partial file exists to resume from
try:
    existing_df = pd.read_csv(OUTPUT_CSV_PATH)
    start_index = len(existing_df)
    print(f"Resuming from index {start_index}.")
except FileNotFoundError:
    existing_df = pd.DataFrame()
    start_index = 0
    print("Starting from scratch.")

# Use tqdm for a progress bar
pbar = tqdm(total=len(sampled_df), initial=start_index)
for i in range(start_index, len(sampled_df)):
    row = sampled_df.iloc[i]

    # --- Generate ---
    model_answer, full_prompt = generate_squad_answer_multi_scenario(model, tokenizer, row)

    # --- Judge ---
    # The judge ALWAYS compares against the original, correct context and answer
    ground_truth_answer = row['answers']['text'][0] if row['answers']['text'] else ""
    label = judge_squad_answer(
        context=row['context'], # Always the original context
        question=row['question'],
        model_answer=model_answer,
        ground_truth_answer=ground_truth_answer,
        api_key=API_KEY
    )

    # Store the result
    results_list.append({
        'scenario': row['scenario'],
        'full_prompt': full_prompt, # The actual text fed to the model
        'model_answer': model_answer,
        'ground_truth_answer': ground_truth_answer,
        'hallucination_label': label,
        'original_context': row['context'], # Keep for reference
        'question': row['question']
    })

    pbar.update(1)

    # --- Save progress in batches and print counts ---
    if (i + 1) % BATCH_SIZE == 0 or (i + 1) == len(sampled_df):
        temp_df = pd.DataFrame(results_list)

        # Append to the existing DataFrame and save
        if not existing_df.empty:
            combined_df = pd.concat([existing_df, temp_df], ignore_index=True)
        else:
            combined_df = temp_df

        combined_df.to_csv(OUTPUT_CSV_PATH, index=False)

        print(f"\nSaved batch up to index {i}. Total rows in file: {len(combined_df)}")

        # Print count of 1s and 0s for the saved batch
        batch_start_index = len(combined_df) - len(temp_df)
        batch_end_index = len(combined_df) - 1
        print(f"Batch {batch_start_index//BATCH_SIZE + 1} (rows {batch_start_index} to {batch_end_index}):")
        valid_labels = temp_df[temp_df['hallucination_label'] != -1]['hallucination_label']
        if not valid_labels.empty:
             counts = valid_labels.value_counts().sort_index()
             print(counts)
        else:
             print("No valid labels in this batch.")


        # Update for next resume
        existing_df = combined_df
        results_list = [] # Clear the list for the next batch

pbar.close()
print("Phase 1 complete. Labeled dataset saved to Google Drive.")

# --- Final check of the class balance ---
final_df = pd.read_csv(OUTPUT_CSV_PATH)
print("\nFinal Class Balance:")
# We filter out any rows where the judge failed (returned -1)
print(final_df[final_df['hallucination_label'] != -1]['hallucination_label'].value_counts())
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", forc
Starting from scratch.

Saved batch up to index 19. Total rows in file: 20
Batch 1 (rows 0 to 19):
hallucination_label
0    18
1     2
Name: count, dtype: int64

Saved batch up to index 39. Total rows in file: 40
Batch 2 (rows 20 to 39):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 59. Total rows in file: 60
Batch 3 (rows 40 to 59):
hallucination_label
0    20
Name: count, dtype: int64

Saved batch up to index 79. Total rows in file: 80
Batch 4 (rows 60 to 79):
hallucination_label
0    18
1     2
Name: count, dtype: int64

Saved batch up to index 99. Total rows in file: 100
Batch 5 (rows 80 to 99):
hallucination_label
0    17
1     3
Name: count, dtype: int64

Saved batch up to index 119. Total rows in file: 120
Batch 6 (rows 100 to 119):
hallucination_label
0    18
1     2
Name: count, dtype: int64

Saved batch up to index 139. Total rows in file: 140
Batch 7 (rows 120 to 139):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 159. Total rows in file: 160
Batch 8 (rows 140 to 159):
hallucination_label
0    17
1     3
Name: count, dtype: int64

Saved batch up to index 179. Total rows in file: 180
Batch 9 (rows 160 to 179):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 199. Total rows in file: 200
Batch 10 (rows 180 to 199):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 219. Total rows in file: 220
Batch 11 (rows 200 to 219):
hallucination_label
0    17
1     3
Name: count, dtype: int64

Saved batch up to index 239. Total rows in file: 240
Batch 12 (rows 220 to 239):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 259. Total rows in file: 260
Batch 13 (rows 240 to 259):
hallucination_label
0    19
```

```
1     1
Name: count, dtype: int64

Saved batch up to index 279. Total rows in file: 280
Batch 14 (rows 260 to 279):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 299. Total rows in file: 300
Batch 15 (rows 280 to 299):
hallucination_label
0    18
1     2
Name: count, dtype: int64

Saved batch up to index 319. Total rows in file: 320
Batch 16 (rows 300 to 319):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 339. Total rows in file: 340
Batch 17 (rows 320 to 339):
hallucination_label
0    18
1     2
Name: count, dtype: int64

Saved batch up to index 359. Total rows in file: 360
Batch 18 (rows 340 to 359):
hallucination_label
0    18
1     2
Name: count, dtype: int64

Saved batch up to index 379. Total rows in file: 380
Batch 19 (rows 360 to 379):
hallucination_label
0    18
1     2
Name: count, dtype: int64

Saved batch up to index 399. Total rows in file: 400
Batch 20 (rows 380 to 399):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 419. Total rows in file: 420
Batch 21 (rows 400 to 419):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 439. Total rows in file: 440
Batch 22 (rows 420 to 439):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 459. Total rows in file: 460
Batch 23 (rows 440 to 459):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 479. Total rows in file: 480
Batch 24 (rows 460 to 479):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 499. Total rows in file: 500
Batch 25 (rows 480 to 499):
hallucination_label
0    20
Name: count, dtype: int64

Saved batch up to index 519. Total rows in file: 520
Batch 26 (rows 500 to 519):
hallucination_label
0    18
1     2
```

```
Name: count, dtype: int64

Saved batch up to index 539. Total rows in file: 540
Batch 27 (rows 520 to 539):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 559. Total rows in file: 560
Batch 28 (rows 540 to 559):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 579. Total rows in file: 580
Batch 29 (rows 560 to 579):
hallucination_label
0    18
1     2
Name: count, dtype: int64

Saved batch up to index 599. Total rows in file: 600
Batch 30 (rows 580 to 599):
hallucination_label
0    20
Name: count, dtype: int64

Saved batch up to index 619. Total rows in file: 620
Batch 31 (rows 600 to 619):
hallucination_label
0    18
1     2
Name: count, dtype: int64

Saved batch up to index 639. Total rows in file: 640
Batch 32 (rows 620 to 639):
hallucination_label
0    16
1     4
Name: count, dtype: int64

Saved batch up to index 659. Total rows in file: 660
Batch 33 (rows 640 to 659):
hallucination_label
0    17
1     3
Name: count, dtype: int64

Saved batch up to index 679. Total rows in file: 680
Batch 34 (rows 660 to 679):
hallucination_label
0    18
1     2
Name: count, dtype: int64

Saved batch up to index 699. Total rows in file: 700
Batch 35 (rows 680 to 699):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 719. Total rows in file: 720
Batch 36 (rows 700 to 719):
hallucination_label
0    20
Name: count, dtype: int64

Saved batch up to index 739. Total rows in file: 740
Batch 37 (rows 720 to 739):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 759. Total rows in file: 760
Batch 38 (rows 740 to 759):
hallucination_label
0    17
1     3
Name: count, dtype: int64

Saved batch up to index 779. Total rows in file: 780
Batch 39 (rows 760 to 779):
hallucination_label
0    19
1     1
Name: count, dtype: int64
```

```
Saved batch up to index 799. Total rows in file: 800
Batch 40 (rows 780 to 799):
hallucination_label
0    16
1     4
Name: count, dtype: int64

Saved batch up to index 819. Total rows in file: 820
Batch 41 (rows 800 to 819):
hallucination_label
0    14
1     6
Name: count, dtype: int64

Saved batch up to index 839. Total rows in file: 840
Batch 42 (rows 820 to 839):
hallucination_label
0    20
Name: count, dtype: int64

Saved batch up to index 859. Total rows in file: 860
Batch 43 (rows 840 to 859):
hallucination_label
0    17
1     3
Name: count, dtype: int64

Saved batch up to index 879. Total rows in file: 880
Batch 44 (rows 860 to 879):
hallucination_label
0    20
Name: count, dtype: int64

Saved batch up to index 899. Total rows in file: 900
Batch 45 (rows 880 to 899):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 919. Total rows in file: 920
Batch 46 (rows 900 to 919):
hallucination_label
0    17
1     3
Name: count, dtype: int64

Saved batch up to index 939. Total rows in file: 940
Batch 47 (rows 920 to 939):
hallucination_label
0    20
Name: count, dtype: int64

Saved batch up to index 959. Total rows in file: 960
Batch 48 (rows 940 to 959):
hallucination_label
0    19
1     1
Name: count, dtype: int64

Saved batch up to index 979. Total rows in file: 980
Batch 49 (rows 960 to 979):
hallucination_label
0    20
Name: count, dtype: int64

Saved batch up to index 999. Total rows in file: 1000
Batch 50 (rows 980 to 999):
hallucination_label
0    18
1     2
Name: count, dtype: int64

Saved batch up to index 1019. Total rows in file: 1020
Batch 51 (rows 1000 to 1019):
hallucination_label
0     3
1    17
Name: count, dtype: int64

Saved batch up to index 1039. Total rows in file: 1040
Batch 52 (rows 1020 to 1039):
hallucination_label
0     3
1    17
Name: count, dtype: int64

Saved batch up to index 1059. Total rows in file: 1060
Batch 53 (rows 1040 to 1059):
hallucination label
```

```
0     5
1    15
Name: count, dtype: int64

Saved batch up to index 1079. Total rows in file: 1080
Batch 54 (rows 1060 to 1079):
hallucination_label
0     5
1    15
Name: count, dtype: int64

Saved batch up to index 1099. Total rows in file: 1100
Batch 55 (rows 1080 to 1099):
hallucination_label
0     5
1    15
Name: count, dtype: int64

Saved batch up to index 1119. Total rows in file: 1120
Batch 56 (rows 1100 to 1119):
hallucination_label
0     1
1    19
Name: count, dtype: int64

Saved batch up to index 1139. Total rows in file: 1140
Batch 57 (rows 1120 to 1139):
hallucination_label
0     2
1    18
Name: count, dtype: int64

Saved batch up to index 1159. Total rows in file: 1160
Batch 58 (rows 1140 to 1159):
hallucination_label
0     5
1    15
Name: count, dtype: int64

Saved batch up to index 1179. Total rows in file: 1180
Batch 59 (rows 1160 to 1179):
hallucination_label
0     4
1    16
Name: count, dtype: int64

Saved batch up to index 1199. Total rows in file: 1200
Batch 60 (rows 1180 to 1199):
hallucination_label
0     3
1    17
Name: count, dtype: int64

Saved batch up to index 1219. Total rows in file: 1220
Batch 61 (rows 1200 to 1219):
hallucination_label
0     1
1    19
Name: count, dtype: int64

Saved batch up to index 1239. Total rows in file: 1240
Batch 62 (rows 1220 to 1239):
hallucination_label
0     5
1    15
Name: count, dtype: int64

Saved batch up to index 1259. Total rows in file: 1260
Batch 63 (rows 1240 to 1259):
hallucination_label
0     5
1    15
Name: count, dtype: int64

Saved batch up to index 1279. Total rows in file: 1280
Batch 64 (rows 1260 to 1279):
hallucination_label
0     7
1    13
Name: count, dtype: int64

Saved batch up to index 1299. Total rows in file: 1300
Batch 65 (rows 1280 to 1299):
hallucination_label
0     2
1    18
Name: count, dtype: int64

Saved batch up to index 1319. Total rows in file: 1320
Batch 66 (rows 1300 to 1319):
hallucination_label
```

```
hallucination_label
0     5
1    15
Name: count, dtype: int64

Saved batch up to index 1339. Total rows in file: 1340
Batch 67 (rows 1320 to 1339):
hallucination_label
0     3
1    17
Name: count, dtype: int64

Saved batch up to index 1359. Total rows in file: 1360
Batch 68 (rows 1340 to 1359):
hallucination_label
0     6
1    14
Name: count, dtype: int64

Saved batch up to index 1379. Total rows in file: 1380
Batch 69 (rows 1360 to 1379):
hallucination_label
0     3
1    17
Name: count, dtype: int64

Saved batch up to index 1399. Total rows in file: 1400
Batch 70 (rows 1380 to 1399):
hallucination_label
0     5
1    15
Name: count, dtype: int64

Saved batch up to index 1419. Total rows in file: 1420
Batch 71 (rows 1400 to 1419):
hallucination_label
0     3
1    17
Name: count, dtype: int64

Saved batch up to index 1439. Total rows in file: 1440
Batch 72 (rows 1420 to 1439):
hallucination_label
0     4
1    16
Name: count, dtype: int64

Saved batch up to index 1459. Total rows in file: 1460
Batch 73 (rows 1440 to 1459):
hallucination_label
0     4
1    16
Name: count, dtype: int64

Saved batch up to index 1479. Total rows in file: 1480
Batch 74 (rows 1460 to 1479):
hallucination_label
0     4
1    16
Name: count, dtype: int64

Saved batch up to index 1499. Total rows in file: 1500
Batch 75 (rows 1480 to 1499):
hallucination_label
0     1
1    19
Name: count, dtype: int64

Saved batch up to index 1519. Total rows in file: 1520
Batch 76 (rows 1500 to 1519):
hallucination_label
0     1
1    19
Name: count, dtype: int64

Saved batch up to index 1539. Total rows in file: 1540
Batch 77 (rows 1520 to 1539):
hallucination_label
0     1
1    19
Name: count, dtype: int64

Saved batch up to index 1559. Total rows in file: 1560
Batch 78 (rows 1540 to 1559):
hallucination_label
0     2
1    18
Name: count, dtype: int64

Saved batch up to index 1579. Total rows in file: 1580
Batch 79 (rows 1560 to 1579):
```