HAUNT SCRIPT

Chembian RK-22BAI1431
Sarveesh Kumar-22BAI1420
Vishwa-22bai1466

Data Collection Plan

Objective: To outline how data, prompts, and inputs will be collected, analyzed, and used to refine model output in the absence of traditional datasets.

Approach: HauntScript does not rely on a pre-compiled dataset since it uses a pre-trained generative model (Gemini 1.5 Flash). Instead, the focus is on crafting, capturing, and refining user inputs (prompts) and observing the generated output for tuning. We collected input-output samples from various test cases to build a pseudo dataset that simulates diverse user scenarios.

Steps Taken:

- Designed 30+ test prompts with varying complexity and emotional tone.
- Collected user inputs through local UI tests and peer usage.
- Stored outputs along with parameters (temperature, token limit) and generation latency.
- Categorized prompts into: atmospheric horror, psychological horror, supernatural horror, and thriller variants.

This data was then annotated for fluency, horror tone, grammar, coherence, and length adherence. These annotations helped us guide prompt formatting in later stages.

Source Identification Report

Objective: Identify and document all sources from which data (if any) or model-based content is derived or interacted with.

Data Origin:

- The language model itself is a hosted solution via Google Gemini's API.
- All prompts are generated dynamically using user inputs collected on the frontend.
- No copyrighted material or pre-written stories are used or stored.

External Dependencies:

- Google Generative AI SDK for prompt submission.
- Python scripts developed internally.

- Open-source tools like Streamlit, Anaconda, and PyDot for visualization.

Ethical Considerations:

- No personal data is collected or logged.
- The system uses ephemeral prompts that are not stored on cloud servers.
- Every prompt follows strict language filters to avoid offensive or dangerous outputs.

Data Quality Report

Goal: Measure the quality, clarity, and emotional impact of AI-generated horror stories across multiple runs.

Process: Each sample output was manually rated by at least two evaluators across 5 parameters:

1. **Relevance to Prompt** (Did it stick to the situation?)
2. **Horror Tone** (Did it evoke tension, fear, or dread?)
3. **Creativity** (Originality in plot or description)
4. **Language Fluency** (Grammar, sentence construction)
5. **Length Accuracy** (Adherence to requested line count)

Observations:

- Average horror tone score: 8.2/10
- Average fluency score: 9.1/10
- Most common issue: overuse of clichés in short responses (<5 lines)
- Strengths: vivid vocabulary, consistent grammar, effective buildup in longer outputs

Sample Insight:

> Input: “A lone girl hears footsteps behind her in a dark forest.”
> Output: “The girl turned, but the shadows whispered instead of revealing a face.
> Each step she took echoed a heartbeat not her own.”
> Evaluation: Highly atmospheric, vivid, concise, and effective.

Preprocessing Report

Goal: Ensure that all user inputs are clean, structured, and context-rich before being sent to the AI model.

Input Handling:

- Used `st.text_input()` to capture `character_name` and `situation`.
- Used `st.number_input()` to limit `number_of_lines` between 1 and 20.
- Applied `.strip()` and `.replace("\n", "")` to sanitize user text inputs.

- Enforced prompt format consistency:

prompt = f"Write a horror story featuring the character '{character_name}' in the situation '{situation}' within {no_of_lines} lines."

Advanced Prompting Techniques:

- Introduced soft constraints like "vividly describe", "use suspense-building phrases", etc.
- Added optional genre modifiers (dark fantasy, gothic horror) during internal testing.
- Limited over-specification to maintain model flexibility.

Validation Rules:

- Must not contain empty fields
- Situation input must be between 10 and 250 characters
- Number of lines must be an integer (validated)

Result: Preprocessing enabled a more consistent output across different users. The standardized format increased alignment between prompt and response and reduced hallucination.

Conclusion: Despite not requiring traditional datasets, the data collection and preprocessing phase was critical in fine-tuning user experience and model output. By systematically analyzing prompt effectiveness and output structure, we created a pseudo-evaluation dataset that helped improve model behavior, storytelling quality, and user engagement. This groundwork led to improved performance in the development and tuning phases.