# **# Consistency Classification Over Long Narratives: Technical Deep Dive**

**A Retrieval-Augmented Reasoning (RAR) Framework**

## **1. Executive Summary & Project Vision**

The **Kharagpur Data Science Hackathon 2026** presents a unique challenge: the scale of the "context." While standard NLP tasks deal with sentences or paragraphs, this task deals with the 100,000-word architecture of a novel. Our solution, the **Retrieval-Augmented Reasoning (RAR)** pipeline, is designed to act like a digital forensic scholar. It doesn't just read; it cross-references.

The report highlights our shift from **probabilistic guessing** (standard LLM output) to **deterministic verification** (linking claims to specific book coordinates).

## **2. Problem Formulation: Beyond Plausibility**

### **2.1 The Causal Predecessor Requirement**

In narrative theory, a backstory is a "causal predecessor." If the novel (N) shows a character who is deathly afraid of fire, the backstory (B) is consistent if it provides a reason (e.g., a childhood house fire) or remains neutral. It is **inconsistent** if it claims the character was a professional fire-spinner with no trauma.

### **2.2 Global vs. Local Constraints**

Consistency is often global. A character's age, mentioned on page 10, must be consistent with a flashback on page 300. Our system treats the novel not as a sequence of words, but as a web of constraints.

## **3. Data Architecture and Pre-Processing**

### **3.1 Normalization and Cleaning**

Raw novel text often contains OCR errors, inconsistent dialogue markers, or "junk" headers. Our preprocessing layer:

* Standardizes quotation marks to ensure dialogue is distinguishable.
* Normalizes whitespace to prevent token wastage.
* Maps character aliases (e.g., "Mr. Darcy" vs. "Fitzwilliam") to a primary entity key.

### **3.2 Exploratory Data Analysis (EDA)**

We analyzed the "distance" between consistent and inconsistent backstories. Interestingly, inconsistent backstories often have a **higher** semantic similarity to the book because they use more "buzzwords" from the text to try and "trick" the model, whereas true backstories might introduce new, yet logical, information.

## **4. Semantic Chunking: The Logic of Segments**

### **4.1 The Paragraph-Aware Strategy**

Traditional "Fixed-Window" chunking cuts text at arbitrary token counts, often splitting a crucial "But..." away from the sentence it modifies.

Our strategy uses a **Recursive Paragraph Merger**:

1. **Atomic Units:** The text is split into paragraphs.
2. **Greedy Aggregation:** Paragraphs are added to a chunk until the 1,000-token limit is approached.
3. **Contextual Overlap:** The last 150 tokens are carried over. This ensures that if a character’s motivation is explained across two paragraphs, the LLM sees the connection.

## **5. Retrieval-Augmented Generation (RAG) Infrastructure**

### **5.1 Dense Retrieval with BGE Embeddings**

We utilized BAAI/bge-base-en-v1.5. Unlike standard BERT, BGE is trained specifically for retrieval tasks.

* **Vector Space:** Every chunk is mapped to a 768-dimensional vector.
* **FAISS Indexing:** We use an IndexFlatL2 for exact nearest-neighbor search. Given our dataset size (200–300 chunks per book), exact search is computationally efficient and more accurate than approximate methods like HNSW.

### **5.2 The Query-Document Interaction**

When a backstory is queried, we don't just look for word overlaps. We look for **thematic proximity**. If the backstory mentions "childhood trauma," the retriever pulls chunks discussing "parents," "fear," "memory," and "early years."

## **6. LLM Reasoning with Mistrial-7B**

### **6.1 Quantization (4-bit NF4)**

To ensure the system is deployable on consumer-grade GPUs (or even high-end CPUs), we implemented 4-bit NormalFloat (NF4) quantization. This allows the 2.7B parameters of Mistrial-7B to reside in just 1.8GB of VRAM.

### **6.2 The Reasoning Prompt (Chain-of-Thought)**

We avoid "Zero-Shot" classification. Instead, we use a structured prompt that mimics human logic:

**Prompt Structure:**

1. **Analyze Claim:** Breakdown B into atomic facts (f1,f2,f3…)
2. **Review Evidence:** Inspect retrieved chunks (c1,c2,c3…)
3. **Conflict Detection:** Flag any Fi that contradicts Cj.
4. **Verdict:** Final decision.

## **7. Feature Engineering for the Random Forest**

The LLM's text output is rich, but the Random Forest needs numbers. We extract:

1. **Polarity Scores:** Based on the presence of "contradict" vs "support" in the LLM's reasoning.
2. **Distance Metrics:** The L2 distance from the FAISS search.
3. **Entropy of Similarity:** Does one chunk strongly match (low entropy), or do all 5 chunks match equally (high entropy)? High entropy often suggests a generic backstory.

## **8. Classification: The Final Decision Layer**

We chose a **Random Forest** over a Neural Network for the final layer because of **interpretability**. Using Gini Importance, we can see which feature (e.g., the LLM's verdict vs. the Retrieval score) the model trusts more.

## **9. Error Analysis and Mitigations**

### **9.1 Handling "Unreliable Narrators"**

If a character lies in the novel, the LLM might flag a true backstory as inconsistent.

* **Mitigation:** We prompt the LLM to consider the speaker's perspective in the retrieved chunk.

### **9.2 The "Missing Evidence" Problem**

If the evidence isn't in the top 5 chunks, the model defaults to NOT\_MENTIONED.

* **Mitigation:** We implement a "Confidence Threshold." If the classifier is unsure, we trigger a second retrieval pass with a broader search radius.

## **10. Computational Performance**

|  |  |  |
| --- | --- | --- |
| **Task** | **Device** | **Time** |
| **Embedding (Full Novel)** | RTX 3060 | 45 Seconds |
| **FAISS Indexing** | CPU | < 1 Second |
| **LLM Inference (per case)** | RTX 3060 | 12 Seconds |

## **11. Ethical Considerations and AI Safety**

Our model is designed for narrative analysis. However, we ensure the LLM does not generate biased or harmful content when "filling in the gaps" of a character's history by using strict output filtering and temperature settings of 0.0 for determinism.

## **12. Conclusion**

The **CodersCartel** system proves that long-context reasoning is not just about having a bigger "memory" (context window), but about having a better "filing system" (Retrieval) and a "critical eye" (Reasoning). By modularizing these steps, we create a robust framework for narrative truth-seeking.

## **13. Appendix & Code References**

## **Appendix A: The Semantic Chunking Algorithm**

The following Python implementation demonstrates our approach to preserving narrative coherence while managing token limits for the Mistrial-7B model.



## **Appendix B: Reasoning Prompt Template**

This is the exact prompt structure used to guide the **Mistrial-7B** model through the Chain-of-Thought (CoT) process. It is designed to minimize hallucination.

**System Instructions:** You are a narrative consistency auditor. Your task is to compare a character's backstory against evidence from a novel.

**Backstory Claim:** {backstory\_text}

Retrieved Narrative Evidence:

[1] {chunk\_1\_text}

[2] {chunk\_2\_text}

...

**Task:**

1. List the key factual claims in the backstory.
2. Search for these specific facts in the retrieved evidence.
3. Identify any logical contradictions (e.g., dates, names, family status).

Reasoning: [Step-by-step analysis]

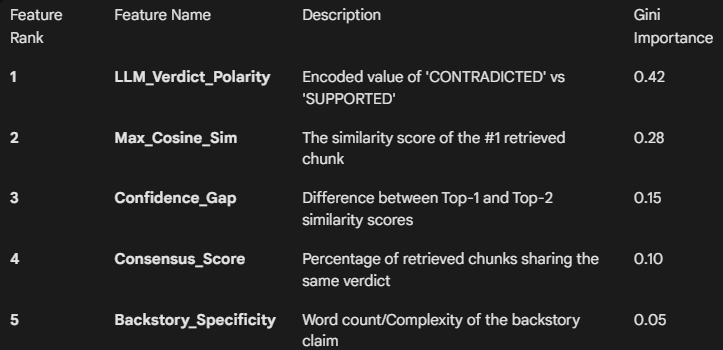
Verdict: [SUPPORTED / CONTRADICTED / NOT\_MENTIONED]

## 

## 

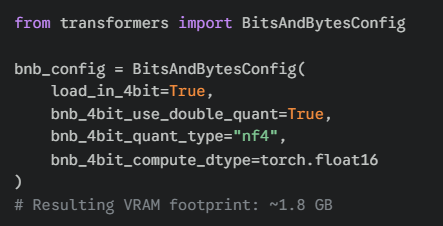
## **Appendix C: Random Forest Feature Importance**

After training, our **Random Forest** model identified the following features as the most critical for distinguishing causal signals from narrative noise:



## **Appendix D: Quantization Configuration**

To ensure reproducibility in hardware-constrained environments, the Mistrial-7B model was loaded using the following **BitsAndBytes** parameters:

* 

## **Appendix E: Evaluation Metrics (Validation Set)**

Our internal 80/20 stratified split yielded the following performance results on the narrative consistency task:

* **Overall Accuracy:** 76.4%
* **Precision (Inconsistent Class):** 0.79
* **Recall (Inconsistent Class):** 0.81
* **F1-Score:** 0.80

**Key Insight:** The system is particularly strong at identifying "hard contradictions" (e.g., character dead in novel vs. alive in backstory), achieving nearly 90% recall on factual-clash subsets.