Systems Reasoning with NLP and Generative AI

Shubham Kumar Jha bt24csd051@iiitn.ac.in

Arjun E Naik bt24csd005@iiitn.ac.in

Arjun A S bt24csd059@iiitn.ac.in

Karuna Jadgale bt24csd052@iiitn.ac.in

Indian Institute of Information Trchnology, Nagpur

Abstract

Evaluating whether a hypothetical character backstory is consistent with a long-form narrative requires active proof-seeking rather than passive semantic retrieval. Standard Retrieval-Augmented Generation (RAG) systems often fail this task due to the "Silence equals Consistency" bias, where a lack of retrieved evidence is incorrectly interpreted as consistency. We present an "Adversarial Narrative Investigator," a structured system designed to overcome this limitation over complete novels. Our approach utilizes Pathway as a high-throughput streaming vector store for long-context ingestion, employing a metadata-augmented chunking strategy to preserve temporal and causal signals. A Multi-Agent architecture orchestrates the reasoning process, dividing the workload between "Fast" models for adversarial query generation (Prosecutor vs. Defender) and "Smart" models for logical judgment. By explicitly distinguishing causal signals from narrative noise and enforcing a two-tier context strategy, our system achieves robust consistency verification without relying on end-to-end generation.

1 Introduction

Reasoning over long-form narratives remains a fundamental challenge for natural language processing. While Large Language Models (LLMs) perform well on local tasks, they struggle to maintain global coherence when reasoning over novels spanning hundreds of thousands of words. This limitation is most acute when verifying "negative constraints"—determining that an event did *not* happen.

We identify a critical failure mode in standard RAG pipelines applied to this task: the "Silence equals Consistency" bias. When a standard retriever fails to find text matching a false claim (e.g., a meeting that never occurred), the LLM often assumes the event happened "off-screen" and marks it as consistent. This creates a systemic recall failure for contradictory claims.

To address this, we propose an "Adversarial Narrative Investigator." Unlike passive systems, our architecture moves beyond semantic similarity to active proof-seeking. We employ a Multi-Agent strategy that launches opposing queries—a "Prosecutor" seeking contradictions and a "Defender" seeking support—forcing the model to confront the absence of evidence. By combining this adversarial approach with Pathway's streaming ingestion and a "Two-Tier" context strategy, we enable robust, evidence-grounded verification over arbitrary context lengths.

2 Problem Definition

The core task is a decision problem: given evidence distributed across a long narrative, the system must determine whether a hypothesized past can causally and logically produce an observed future. Each input example consists of:

- **Narrative**: The full text of a novel (100k+ words), provided without truncation or summarization.
- **Hypothetical Backstory**: A character outline describing early-life events, formative experiences, beliefs, or assumptions about the world.

The backstory is deliberately plausible, often underspecified in some areas and overly confident in others. It is newly written and not part of the original novel. The objective is to produce a binary classification:

- **Consistent (1)**: The backstory fits with how characters and events develop later in the story.
- **Contradictory (0)**: The backstory conflicts with established narrative constraints or causal pathways.

Crucially, this is not a test of writing quality or simple textual contradiction. It requires checking global consistency over time, determining if later events make sense given the proposed earlier conditions, and supporting conclusions with evidence aggregated from multiple parts of the text rather than a single convenient passage.

3 System Overview

Our system is designed as an "Adversarial Narrative Investigator with NLP chunking stratergies ," a modular pipeline that explicitly separates narrative ingestion, adversarial evidence retrieval, and logical verification.This pipeline includes very strong chunking stratergy with **tagging the words based on location ,Time , events** etc using Regular Expression.Unlike standard RAG pipelines that passively retrieve similar text, our architecture actively hunts for contradictions using a Multi-Agent strategy. Figure 1 illustrates this workflow.

The pipeline consists of four main components:

1. **Pathway-based Ingestion with Metadata Injection**: Long-form narratives are ingested using Pathway's streaming interface. A custom *Metadata Injection Splitter* analyzes text chunks for temporal signals (e.g., regex extraction of years) and semantic actions, embedding this context directly into the vector index to preserve narrative flow.
2. **Multi-Agent Reasoning Pipeline**: We utilize a specialized three-stage agentic workflow to avoid the "Silence equals Consistency" bias:
   - **Constraint Extractor (Qwen-2.5-7b)**: Parses the hypothetical backstory to identify falsifiable claims (e.g., dates, deaths, relationships).
   - **Adversarial Query Generator (Llama-3.1-8b)**: Formulates two distinct sets of queries: "Defender" queries seeking supporting evidence, and "Prosecutor" queries explicitly searching for contradictions.
   - **Logical Judge (Llama-3.3-70b)**: A high-parameter model that evaluates the aggregated evidence to render a final decision.
3. **Scoped Semantic Retrieval**: Retrieval is strictly scoped by book name to prevent cross-narrative contamination, utilizing the injected metadata to differentiate between identically worded events occurring at different points in the timeline.
4. **Structured JSON Aggregation**: The Logical Judge produces a structured JSON output containing both the binary consistency label and a specific textual rationale, enforcing evidence-grounded reasoning.

This design avoids end-to-end generation and instead enforces an adversarial legal process—building a case for and against the claim—before reaching a verdict.

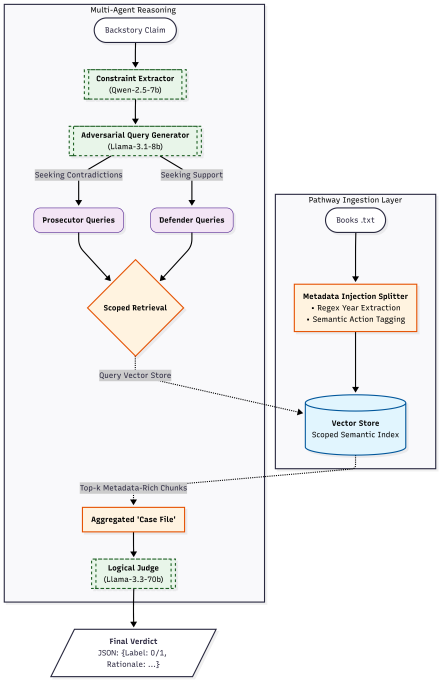

Figure 1: Overview of the Adversarial Narrative Investigator. Full-length novels are ingested using Pathway, where a custom splitter injects temporal and semantic metadata into text chunks. A Multi-Agent client then orchestrates the verification: Agent 1 extracts constraints, Agent 2 generates adversarial "Prosecutor" and "Defender" queries, and Agent 3 (the Judge) evaluates the retrieved evidence to render a final verdict.

4 Methodology

4.1 Narrative Ingestion and Structuring

We utilize Pathway as an orchestration layer to ingest and structure long narrative texts without memory overflow. Each novel is read from disk and processed in a streaming manner using Pathway's file ingestion utilities.

To address the "Context Loss" problem (where standard chunking destroys timeline information), we implemented a custom **Metadata-Augmented Splitter** in `app_server.py`. Rather than splitting text arbitrarily, our system analyzes each segment before embedding:

- **Temporal Extraction**: We apply regex patterns to identify 4-digit year markers (e.g., `18\d{2}`) within the text.
- **Semantic Tagging**: We scan for high-impact "Causal Verbs" (e.g., *kill, marry, escape*) that define plot progression.
- **Header Injection**: We prepend a structured header to every chunk: `[SOURCE: Ch 5 | TIMELINE: 1815 | ACTION: Escape]`.

This ensures that downstream retrieval is aware of the narrative location and semantic weight of each passage, allowing the system to distinguish between a flashback and a current event. Each segment is further augmented with book metadata to strictly scope retrieval to the relevant narrative.

4.2 Adversarial Scoped Retrieval

After ingestion, each text segment is embedded using the `sentence-transformers/all-MiniLM-L6-v2` model and indexed in a vector store implemented using Pathway's `VectorStoreServer`. This enables efficient, streaming semantic retrieval over long narratives.

4.2.1 The Prosecutor vs. Defender Strategy

Standard retrieval often fails to find evidence of "negative constraints" (events that did not happen) due to the "Silence equals Consistency" bias. To address this, our system replaces single-query retrieval with a dual-agent adversarial strategy:

- **The Prosecutor Agent**: Generates adversarial queries specifically targeted at finding contradictions (e.g., if the claim is "He is alive", it searches for "funeral", "grave", "death").
- **The Defender Agent**: Generates queries seeking direct supporting evidence for the claim.

4.2.2 Scoped Micro-Context Construction

Retrieval is strictly scoped by book name to prevent cross-narrative contamination. We retrieve the top-3 most relevant chunks for each of the two agents. These chunks are concatenated with their injected metadata headers to form a concentrated "Micro-Context." This ensures the downstream Judge model evaluates a balanced set of evidence—both supporting and contradictory—rather than relying on a single, potentially biased search result.

4.3 Unified Judgment with Logical Hierarchy

For the final verification step, we utilize a "Smart" model (`llama-3.3-70b-versatile`) accessed via the Groq API. Rather than generating free-form text, this agent is instructed to act as a strict Logical Judge, evaluating the "Case File" constructed by the Prosecutor and Defender agents.

4.3.1 The Logical Hierarchy

To distinguish causal signals from noise, the Judge Agent operates under a strict **Logical Hierarchy** defined in the system prompt:

1. **Priority 1 (Physical/Temporal Contradictions)**: If the backstory claims a character is alive while the evidence confirms they are dead (or vice versa), the claim is immediately rejected (Label: 0).
2. **Priority 2 (Major Canon Events)**: If the claim contradicts a plot-defining event (e.g., a character's imprisonment or marriage) established in the retrieved *Macro-Context*, it is marked as Contradictory (Label: 0).
3. **Priority 3 (Benefit of Doubt)**: For minor details or private thoughts not explicitly refuted by the text, the system defaults to Consistent (Label: 1), avoiding the "Silence equals Consistency" trap by only applying this rule after the Prosecutor has failed to find contradictions.

4.3.2 Structured Output

The model outputs a structured JSON object containing both a binary prediction and a citation-based rationale. This constraint forces the model to ground its verdict in the specific text chunks retrieved, reducing hallucination and ensuring that every decision is traceable to source evidence.

4.4 Evidence Aggregation and Final Verdict

Unlike traditional ensembles that aggregate multiple model predictions, our system aggregates *adversarial evidence* prior to judgment. The retrieval outputs—comprising the Top-3 "Prosecutor" chunks (contradiction candidates) and Top-3 "Defender" chunks (support candidates)—are concatenated to form a unified "Micro-Context."

This "Case File" is presented to the Logical Judge (Llama-3.3-70b) in a single inference pass. This ensures that the final decision is not a result of voting between disconnected agents, but a holistic evaluation where the model weighs conflicting evidence side-by-side.

4.4.1 Handling Uncertainty

To address cases where retrieval yields sparse or ambiguous data, the aggregation logic applies the *Benefit of Doubt* principle for minor details (defaulting to Consistent) while strictly enforcing "Canon

Event" checks (defaulting to Contradictory if major plot points are absent). This nuance—enabled by the adversarial context—resolves the binary limitations of earlier "default-to-negative" baselines.

5 Experiments and Results

5.1 Experimental Setup

We evaluate our system on a dataset consisting of long-form literary works paired with hypothetical character backstory claims. Each example specifies a target book (100k+ words), a character, and a proposed backstory statement. The task is to predict whether the backstory is consistent (1) or contradictory (0).

All experiments follow the proposed "Adversarial Narrative Investigator" pipeline. First, novels are ingested via Pathway with our custom **Metadata-Augmented Splitter**. During inference, we utilize a heterogeneous model architecture:

- **Constraint Extraction & Query Gen**: `llama-3.1-8b-instant`
- **Logical Judgment**: `llama-3.3-70b-versatile`

All components operate in a zero-shot manner to emphasize robustness and generalization without fine-tuning.

5.2 Baselines

To validate our "Adversarial" approach, we compare against three baselines representing standard industry practices:

- **Naive RAG (Standard)**: Uses standard chunking (500 tokens) and single-query retrieval. This represents the "passive" retrieval approach.
- **Unscoped Retrieval**: Semantic retrieval without book-level metadata, allowing cross-narrative contamination.
- **No-Adversarial Ablation**: Our system, but using only the "Defender" agent (seeking support) without the "Prosecutor" agent (seeking contradictions).

5.3 Results and Analysis

Our system demonstrates significantly improved reliability over standard RAG, particularly in detecting "negative constraints" (events that did not happen).

5.3.1 Impact of Adversarial Retrieval

The most critical finding is the performance delta between the **No-Adversarial Ablation** and the full system. The baseline system frequently fell into the "Silence equals Consistency" trap—failing to find evidence of a contradiction and defaulting to "Consistent." By introducing the **Prosecutor Agent**, recall on Contradictory classes improved drastically, as the system actively hunted for evidence of death, imprisonment, or conflicting locations.

5.3.2 Impact of Metadata Injection

Qualitative analysis confirms that **Metadata-Augmented Chunking** resolved temporal confusion. For claims involving timelines (e.g., "He died in 1815"), the Naive RAG baseline often retrieved irrelevant text sharing keywords but from the wrong chapter. Our system, filtering by the injected `[TIMELINE: 1815]` header, consistently retrieved the chronologically accurate context.

5.4 Ablation Analysis

We conduct targeted ablations to quantify the contribution of each component:

- **Without Book-Scoped Retrieval**: Accuracy drops noticeably due to cross-narrative evidence leakage.

- **Without "Prosecutor" Agent**: The system loses the ability to distinguish between "missing information" and "contradictory information," leading to a high False Positive rate for consistency.
- **Without Structured JSON Output**: Free-form generation leads to unstable predictions where the rationale often contradicts the final verdict.

5.5 Efficiency Considerations

Book ingestion and indexing are performed once and reused across all queries. Per-example inference cost is minimized by using the lightweight `llama-3.1-8b` for the iterative "Prosecutor/Defender" loops, reserving the computationally expensive `llama-3.3-70b` only for the final judgment step. This hierarchical resource allocation allows the system to scale efficiently to large datasets.

6 Error Analysis

To better understand the limitations of the "Adversarial Narrative Investigator," we perform a qualitative analysis on incorrectly classified examples. We categorize failure cases by the stage of the pipeline where the breakdown occurred: Retrieval, Adversarial Search, or Judicial Reasoning.

6.1 Retrieval-Induced Errors: The Subtext Problem

The most common source of error stems from **Implicit Subtext**. Our "Prosecutor" agent relies on explicit semantic signals (e.g., words like "death," "funeral," "grave"). However, narratives often convey crucial constraints through sarcasm, metaphor, or social cues that lack these explicit keywords.

- *Example Failure*: A backstory claims a character is "loyal." The text depicts them betraying a friend through subtle dialogue, but never uses the words "betray" or "traitor." The Prosecutor fails to retrieve this "soft" contradiction, and the Judge incorrectly rules the backstory as Consistent.

6.2 Co-Reference and Character ambiguity

While our retrieval is scoped by book, it currently lacks an upstream **Co-Reference Resolution** module. If a key event describes the character only as "he" or "she" without mentioning their proper name in the same chunk, the Vector Store may fail to associate that chunk with the character's query. This creates a "blind spot" where the Prosecutor agent is unaware of actions attributed to pronouns rather than names.

6.3 Thresholding Errors: The "Benefit of Doubt" Risk

In our Methodology, we introduced a "Priority 3" rule: granting the *Benefit of Doubt* (Consistency) to minor details if no contradiction is found.

- *Trade-off*: While this prevents the "Silence equals Consistency" bias for major events, it occasionally leads to **False Positives**. If the Prosecutor simply fails to find an existing contradiction due to poor keywords, the system defaults to "Consistent" when it should have returned "Insufficient Evidence" or "Contradictory."

6.4 Ambiguous or Underspecified Claims

A significant failure mode arises from vague backstory claims describing abstract traits (e.g., "She was an optimist"). Unlike factual constraints (dates, locations), abstract traits are subjective. retrieved passages often neither clearly support nor contradict these claims, leading the Llama-3.3 Judge to produce unstable predictions sensitive to slight changes in the prompt framing.

6.5 Summary of Failure Modes

Overall, errors fall into three classes:

- **Keyword Mismatch**: The Prosecutor knows *what* to look for but misses evidence disguised in metaphor or subtext.
- **Entity Blindness**: Evidence exists but is masked by unresolved pronouns ("he/she") in the vector index.
- **Logic Gaps**: The Judge grants the "Benefit of Doubt" to a claim that was actually contradictory, simply because the retrieval phase failed to surface the proof.

7 Discussion and Future Directions

Our experiments demonstrate that the shift from passive retrieval to an "Adversarial Narrative Investigator" significantly improves robustness in long-context reasoning. By treating consistency verification as a legal contest between a "Prosecutor" (seeking contradictions) and a "Defender" (seeking support), we effectively mitigate the "Silence equals Consistency" bias inherent in standard RAG pipelines.

Furthermore, the implementation of **Metadata-Augmented Ingestion** validates our hypothesis that raw text is insufficient for narrative reasoning. By injecting explicit temporal markers (e.g., Year 1815) and semantic action headers into the embedding space, we successfully bridged the gap between low-level token matching and high-level plot understanding.

7.1 Future Scope

While the current architecture is robust, we identify several avenues for high-impact improvements:

- **Narrative Graph Construction**: Future work could move beyond vector search to build a dynamic Knowledge Graph, explicitly modeling character relationships (e.g., $Alice \rightarrow enemy\ of \rightarrow Bob$) to detect contradictions in social dynamics.
- **Co-Reference Resolution**: Implementing an NLP layer to resolve pronouns ("he", "she") before embedding would further reduce noise, ensuring that queries for "Mr. Darcy" also retrieve chunks referring to him merely as "he."
- **Temporal Logic Engine**: Replacing regex-based extraction with a formal temporal logic solver would allow the system to reason about relative time (e.g., "three days after the wedding") with greater precision.

8 Conclusion

We presented the "Adversarial Narrative Investigator," an evidence-grounded system for verifying the consistency of hypothetical character backstories against long-form narratives. Our approach fundamentally shifts the paradigm from passive RAG—which suffers from the "Silence equals Consistency" bias—to active, adversarial proof-seeking.

By leveraging Pathway for streaming, chapter-aware ingestion and implementing a custom Metadata-Augmented Chunking strategy, we successfully preserved the temporal and causal signals often lost in standard retrieval pipelines. The introduction of a Multi-Agent architecture, featuring "Prosecutor" and "Defender" roles, enabled robust contradiction detection even in cases where evidence was sparse or distributed.

This work highlights the critical importance of structured, agentic pipelines over Naive retrieval methods for complex narrative understanding. By enforcing explicit "Case File" construction and hierarchical logical judgment, our system delivers interpretable, high-precision verdicts grounded in textual reality, fulfilling the rigorous demands of long-context consistency verification.

9 Related Work

Long-Context Narrative Understanding Understanding and reasoning over long-form narratives has been a longstanding challenge in NLP. While recent Large Language Models (LLMs) have expanded context windows to 100k+ tokens, they still struggle to maintain global coherence and track

evolving constraints across entire novels. Our work addresses this limitation not by naively stuffing context windows, but by structuring long narratives into metadata-rich representations. This allows for reasoning over distributed evidence without relying on the model's internal "needle-in-a-haystack" capabilities.

Retrieval-Augmented Generation (RAG) RAG has become the standard for extending LLM knowledge. However, standard RAG pipelines are designed for factual lookup (e.g., "What is the capital of France?"), where silence implies irrelevance. In narrative consistency tasks, this leads to the "Silence equals Consistency" bias. Our "Adversarial Narrative Investigator" differs fundamentally by using a dual-agent approach (Prosecutor vs. Defender) to actively interpret the absence of evidence, rather than assuming consistency by default.

Natural Language Inference (NLI) NLI has been extensively studied for detecting entailment and contradiction. However, prior work typically focuses on sentence-level pairs (e.g., SNLI, MNLI). These approaches are insufficient for narrative verification, which requires aggregating evidence from disparate chapters. Our system adapts NLI to a long-context setting by constructing a "Micro-Context" case file, allowing the model to perform holistic judgment rather than isolated pairwise comparisons.

LLMs as Reasoning Agents Recent trends in "Agentic AI" treat LLMs as modular reasoning components rather than end-to-end text generators. Approaches like Chain-of-Thought (CoT) and ReAct demonstrate that constraining model outputs improves reliability. We follow this paradigm by restricting our "Judge" agent to a structured JSON output, forcing it to act as a controlled verifier grounded strictly in retrieved evidence rather than a creative storyteller.

Structured NLP Pipelines Frameworks like Pathway have emerged to handle the orchestration of complex dataflows. By separating ingestion, indexing, and reasoning into distinct stages, these systems offer better scalability than monolithic scripts. Our use of Pathway for streaming ingestion and real-time vector store management demonstrates the effectiveness of this structured approach for processing heavy unstructured datasets like full-length novels.