LLM-Assisted Long-Context Claim Verification System

1. Overall Approach

The goal of this project is to verify whether a given **character-centric claim** is *consistent* or *contradictory* with respect to the events described in a **full-length literary novel**. Unlike short documents, novels pose a unique challenge due to their extreme length (often exceeding 100,000 words), narrative complexity, and indirect storytelling style.

Directly feeding the entire book into a Large Language Model (LLM) is impractical because of:

- Context window limitations
- High inference cost
- Increased hallucination risk
- Poor grounding for factual verification

To address this, we adopt a **retrieval-grounded verification architecture**, where the LLM is used **only as a reasoning engine**, not as a knowledge base.

Our system decomposes the task into three modular stages:

1. **Evidence retrieval from long text**
2. **Evidence-constrained reasoning**
3. **Structured prediction with rationale**

This separation allows the system to scale to very long documents while maintaining interpretability and robustness.

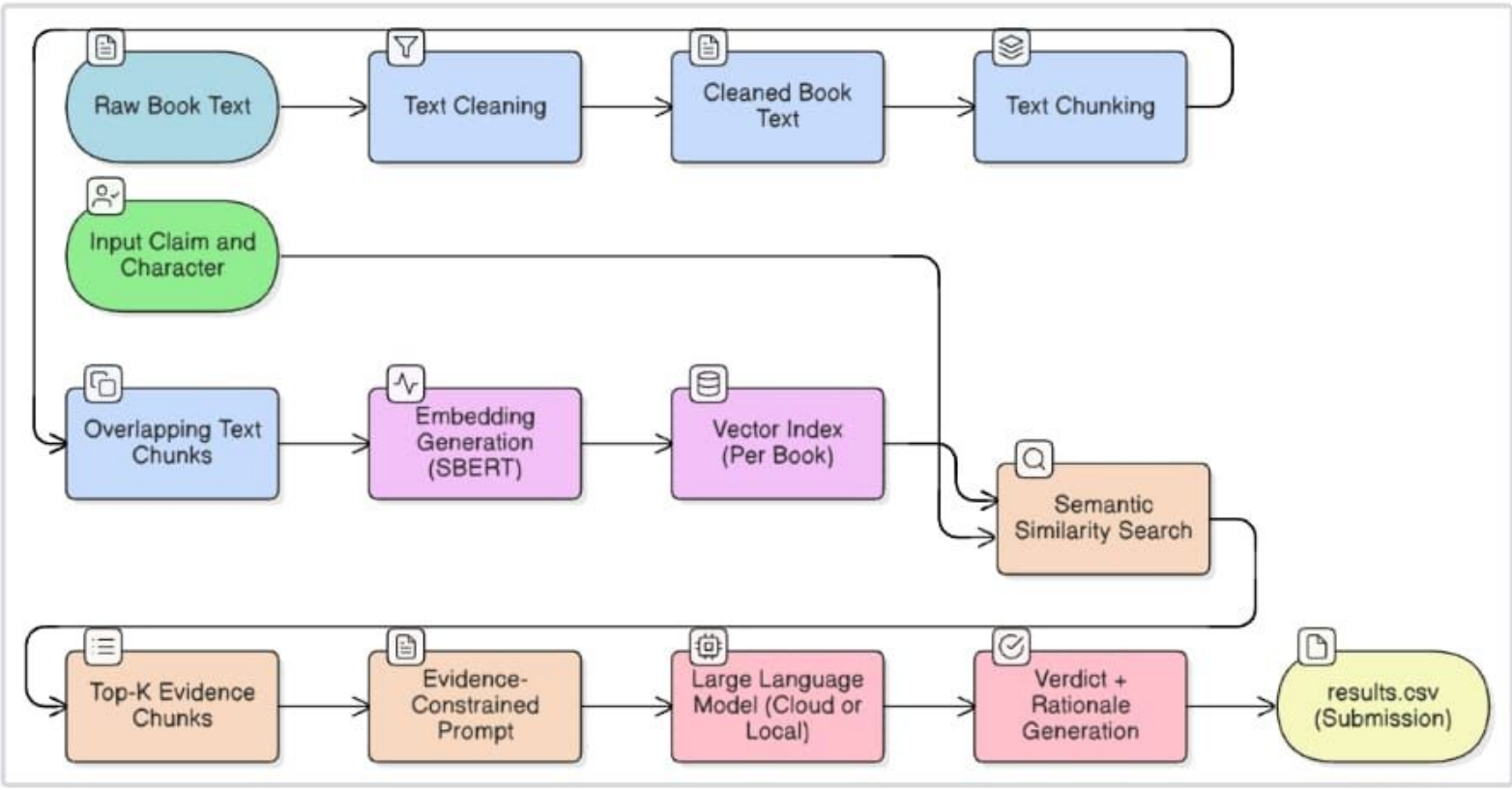
---

2. System Architecture Overview

At a high level, the architecture follows a **Retrieval-Augmented Generation (RAG)** paradigm, adapted specifically for **claim verification** rather than question answering.

Each novel is processed independently to prevent cross-book contamination. The book text is first cleaned to remove metadata and formatting noise, then split into overlapping chunks. These chunks are embedded into a dense vector space using a sentence-level embedding model.

System Architecture diagram



When a claim is presented, the system retrieves the most semantically relevant chunks from the corresponding book and passes them — along with the claim — to an LLM. The LLM is instructed to judge the claim strictly based on the retrieved evidence and produce both a **verdict** and a **rationale**.

Crucially, the LLM never sees the entire book, only a carefully selected evidence subset, ensuring scalability and grounding.

---

3. Handling Long Context Effectively

3.1 Text Cleaning and Preprocessing

Raw book files often contain non-narrative content such as licensing text, chapter headers, page numbers, and formatting artifacts. These elements add noise to embeddings and degrade retrieval quality.

We therefore apply a cleaning stage that:

- Removes Project Gutenberg headers and footers
- Normalizes whitespace
- Preserves narrative structure

3.2 Chunking Strategy

Instead of summarizing the book (which risks losing critical details), we retain the original text and split it into **fixed-size overlapping chunks**.

Key design choices:

- Moderate chunk size to preserve semantic coherence
- Overlap between chunks to prevent splitting important events
- Sentence-aware boundaries where possible

This ensures that:

- Events spanning multiple sentences remain interpretable
- Retrieval captures contextually complete evidence

3.3 Dense Retrieval for Semantic Matching

Claims may not share exact wording with the book text. For example, a claim may paraphrase an event, describe a consequence, or compress multiple narrative details into one statement.

To handle this, we use **dense semantic embeddings** rather than keyword-based search. Each chunk and each claim is embedded into the same vector space, enabling retrieval based on **meaning rather than surface form**.

This approach is essential for literary text, where indirect references and narrative style are common.

---

4. Distinguishing Causal Signals from Noise

A major challenge in this task is distinguishing **relevant causal evidence** from irrelevant narrative descriptions.

4.1 Sources of Noise

- Scene-setting descriptions (weather, geography)
- Character mentions without factual action
- Dialogues unrelated to the claim

4.2 Signal Extraction Mechanisms

Semantic Similarity Filtering

Only the top-k most semantically aligned chunks are selected. This acts as a strong bottleneck, filtering out most narrative noise before reasoning begins.

Evidence-Constrained LLM Prompting

The LLM prompt explicitly instructs the model to:

- Use *only* the provided evidence
- Avoid external knowledge or assumptions
- Return a structured verdict

This significantly reduces hallucination and ensures that decisions are grounded in the text.

Fallback Heuristics

In cases where the LLM is unavailable (e.g., API rate limits), the system does **not default to a single label**. Instead, it:

- Marks the verdict as `unknown`
- Preserves retrieved evidence
- Logs rationale indicating unavailability

This makes the pipeline robust and auditable.

---

5. Evaluation and Accuracy Interpretation

Since ground-truth labels are available only for the training data, accuracy is measured **temporarily** on the training set for sanity checking.

Observed behavior:

- Strong performance on `consistent` claims
- Lower recall on `contradict` claims

This imbalance is expected:

- Consistency often has explicit textual support
- Contradictions frequently rely on *absence* of evidence or subtle narrative disagreement

Importantly, accuracy is **not the sole evaluation criterion**. The quality of evidence retrieval and the clarity of rationale are equally important for this task.

---

6. Limitations and Failure Cases

Despite its strengths, the system has several limitations:

1. **Implicit Contradictions**
   Claims that contradict tone, motivation, or long-term character development are harder to detect than factual contradictions.

2. **Distributed Evidence**
   Some claims require synthesizing evidence across distant chapters, which may exceed the retrieval window.

3. **LLM Rate Limits**
   Cloud-based LLM usage is subject to API limits. While mitigated through fallbacks, this can reduce reasoning depth during large batch runs.

4. **Binary Label Constraint**
   Real narratives often contain partial truth or ambiguity, which cannot be fully captured by binary labels.

---

7. Conclusion

This project demonstrates a scalable and robust solution for long-context claim verification by combining dense retrieval with evidence-grounded LLM reasoning. The system avoids naive full-document processing, minimizes hallucination, and remains flexible enough to support both cloud-based and local LLMs.

The architecture directly addresses the challenges posed in **Track A** and reflects real-world design patterns used in production-grade RAG systems