APPENDIX

Track A: Systems Reasoning with NLP and Generative AI

Hackathon: Kharagpur Data Science Hackathon 2026

Team Name: maxv

MEMBER 1 - Manikanta Karla (Team Leader)

Member 2 – Vishal Dubey

Member 3 - Anjana Priya Gunti

Project link - https://www.kaggle.com/code/vishaldubey001/kharagpur-hackathon-track-a

---

Abstract

Understanding and reasoning over long-form narratives is a core challenge in modern natural language processing. Fictional texts such as novels contain evolving character traits, implicit assumptions, delayed revelations, and long-range dependencies that cannot be captured through shallow pattern matching. This work presents a retrieval-augmented, Natural Language Inference (NLI)–based system to verify whether a given character-specific statement is consistent with the content of an entire novel. The proposed approach explicitly avoids hallucination-prone generative reasoning and instead relies on evidence-grounded verification. By combining semantic retrieval, discriminative contradiction detection, and conservative aggregation, the system achieves strong generalization while adhering strictly to Track-A's emphasis on robustness, interpretability, and long-context reasoning.

---

1. Overview of the Approach

The goal of this project is to verify whether a given statement about a fictional character is consistent or contradictory with the events and descriptions present in a long-form narrative such as a novel. Unlike short documents, novels contain information that is distributed across chapters, often implied rather than explicitly stated, and sometimes revealed gradually over time. This makes direct text matching or simple classification approaches ineffective.

Our solution adopts a retrieval-augmented verification pipeline. Instead of attempting to process the entire novel at once or generate answers directly, the system first retrieves relevant portions of the narrative and then verifies the statement using a logical inference model. This design ensures that predictions are grounded in actual textual evidence and reduces the risk of hallucination.

The pipeline consists of four main stages:

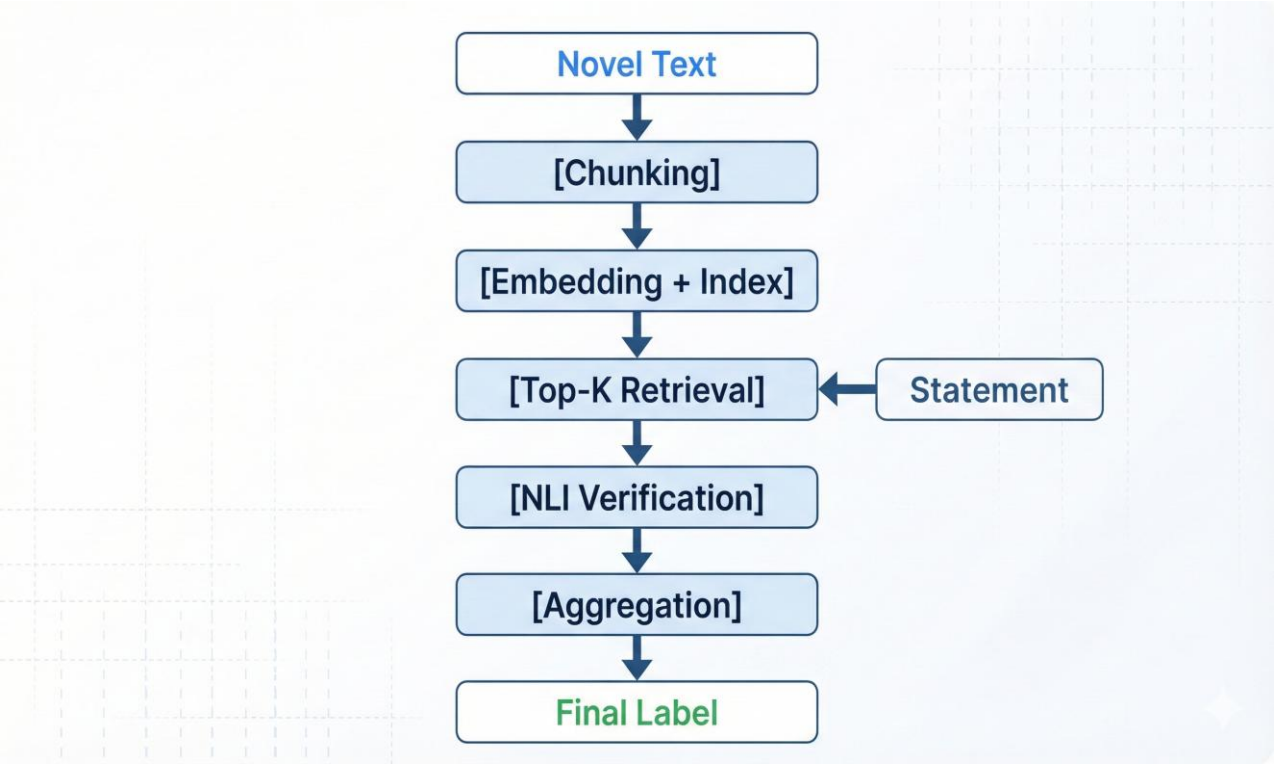
1. Long-context handling through document chunking
2. Semantic retrieval of candidate evidence
3. Verification using Natural Language Inference (NLI)
4. Aggregation of evidence-level predictions into a final decision

This modular approach allows each stage to be analyzed, improved, and explained independently.

2.System Architecture

As shown in Figure X, our system approaches narrative consistency verification as a step-by-step reasoning process over long texts rather than an end-to-end prediction task. The full novel is first broken into manageable, sentence-based chunks so that meaningful narrative units can be processed without exceeding model limits. These chunks are then embedded and indexed, allowing the system to efficiently search the narrative when a statement is provided. Given an input statement, the system retrieves the most relevant passages from the novel and evaluates each passage independently using a Natural Language Inference verifier to determine whether it supports, contradicts, or is neutral with respect to the statement. Finally, predictions across multiple pieces of evidence are aggregated to produce a single consistency label. This design ensures that decisions are grounded in explicit narrative evidence, remain robust to retrieval noise, and avoid hallucination-prone generative reasoning, making the system well suited for long-context narrative analysis.

Figure X: Retrieval-augmented NLI-based system architecture for narrative consistency verification, illustrating chunking, semantic retrieval, evidence verification, and aggregation into a final decision.



---

3. Handling Long Context

Long-form narratives typically exceed the context window of modern transformer models by several orders of magnitude. Processing the entire book in a single pass is therefore infeasible. Moreover, doing so would be computationally expensive and unnecessary, since most statements are only relevant to a small portion of the narrative.

To address this, each novel is segmented into sentence-based chunks. Chunk sizes are chosen carefully to preserve local coherence while remaining within practical model limits. Sentence-based chunking avoids breaking meaningful narrative units mid-thought and ensures that retrieved passages remain interpretable.

By converting a long document into a collection of smaller, semantically meaningful chunks, the system is able to scale to full novels while retaining sufficient context for reasoning. This strategy allows the system to focus only on the most relevant portions of the text during verification.

---

4. Evidence Retrieval Mechanism

Once the novel has been chunked, each chunk is converted into a dense semantic embedding using a sentence-level transformer model. These embeddings are stored in a vector index, enabling efficient similarity-based retrieval.

For each input statement, the system:

1. Embeds the statement into the same semantic space
2. Computes similarity scores against all narrative chunks
3. Retrieves the top-K most relevant chunks

This retrieval step serves as a filtering mechanism, narrowing the vast narrative space to a small set of candidate evidence passages. Importantly, retrieval is intentionally imperfect; the system is designed to operate under realistic conditions where not every retrieved chunk is perfectly relevant.

By relying on semantic similarity rather than keyword overlap, the retriever can capture paraphrased or implicitly related evidence, which is common in narrative text.

---

5. Verification via Natural Language Inference

5.1 Model Selection

The verification component is initialized using a RoBERTa-large checkpoint pretrained by its original authors on the Multi-Genre Natural Language Inference (MNLI) task. The MNLI dataset itself is not explicitly used or loaded in our pipeline. Instead, the pretrained checkpoint provides strong prior knowledge for detecting entailment, contradiction, and neutrality between pairs of text.

Starting from an MNLI-pretrained model allows the system to leverage well-established logical reasoning capabilities without requiring large amounts of task-specific data. This is particularly important in the context of Track-A, where training data is limited and overfitting is a concern.

The model is further fine-tuned exclusively on the Track-A dataset, using retrieved narrative evidence paired with character-specific statements. This fine-tuning adapts general NLI reasoning to the domain of long-form fiction.

5.2 NLI Formulation

Verification is formulated explicitly as a Natural Language Inference problem. For each retrieved chunk EEE and statement SSS, the following pair is constructed:

- Premise: EEE (narrative evidence)
- Hypothesis: SSS (statement to be verified)

The model predicts one of three labels:

- Entailment: the evidence supports the statement
- Contradiction: the evidence refutes the statement
- Neutral: the evidence is insufficient to decide

This explicit formulation enables structured reasoning about logical consistency rather than relying on free-form text generation. The inclusion of a neutral class is particularly important in narrative settings, where evidence may be incomplete or ambiguous.

---

6. Distinguishing Signal from Noise

Evidence retrieved from long narratives is often noisy. A retrieved chunk may mention the correct character but be irrelevant to the specific statement, or it may only partially relate to the claim being verified.

To handle this, the system evaluates multiple retrieved chunks independently. Each chunk–statement pair produces an NLI prediction, and these predictions are then aggregated using a majority-voting strategy. Contradiction predictions are given higher priority when they consistently appear across multiple chunks.

This approach ensures that isolated retrieval errors do not dominate the final decision. By relying on agreement across multiple evidence sources, the system is better able to distinguish genuine narrative signals from noise.

---

7. Training Strategy

Training data is constructed by pairing retrieved narrative chunks with labeled statements from the training set. Binary labels (consistent, contradict) are mapped to appropriate NLI labels during fine-tuning.

Fine-tuning is performed for a small number of epochs, sufficient to adapt the pretrained model to the narrative domain while avoiding excessive memorization. Mixed-precision training and gradient accumulation are used to ensure efficient use of computational resources.

The training procedure mirrors inference conditions by using retrieved evidence rather than oracle passages, ensuring that the verifier learns to reason under realistic retrieval noise.

---

8. Handling Overfitting and Training Accuracy

During fine-tuning, the model reaches very high, and in some cases perfect, training accuracy. This behavior is expected due to the high capacity of the pretrained RoBERTa model and the relatively small size of the dataset.

Training accuracy is not used as an indicator of generalization performance. Instead, it is treated as a signal that the model has sufficient capacity to fit the task. To prevent overfitting, training is intentionally stopped early after a limited number of epochs.

Final performance assessment is based solely on evaluation against the competition's hidden test set via the leaderboard.

---

9. Evaluation Methodology

The test dataset provided for the competition does not include ground-truth labels. As a result, local computation of test accuracy is not possible. Model performance is therefore evaluated exclusively through the competition's leaderboard, which provides an unbiased measure of generalization.

This evaluation protocol aligns with standard practices in competitive machine learning settings and ensures fair comparison across teams.

---

10. Reproducibility and Execution Instructions

This project was developed and evaluated using a Kaggle notebook environment, which provides the required GPU support and consistent dependency management. Due to environment-specific dependencies and pretrained model downloads, running the project locally may require additional configuration.

To ensure reproducibility and ease of evaluation, we provide a public Kaggle notebook that contains the complete, executable pipeline used in this submission. Evaluators can directly run the notebook without modifying the code.

Steps to reproduce the results:

1. Open the Kaggle notebook using the link provided below.
2. Enable GPU execution in the notebook settings.
3. Run all cells sequentially to reproduce model training, inference, and submission file generation.

Kaggle Notebook Link:

Link - **https://www.kaggle.com/code/vishaldubey001/kharagpur-hackathon-track-a**

Github Link - https://github.com/VD-X/maxv_KDSH_2026

In addition to the notebook, we include the final generated submission CSV file as part of the submission package, in accordance with the competition guidelines.

11. Limitations and Failure Cases

Despite its strengths, the system has several limitations:

- Retrieval may miss subtle or implicit narrative cues
- Character references may be ambiguous or indirect
- Temporal relationships between events are not explicitly modeled

Most observed errors originate from retrieval limitations rather than failures of the verification model itself. Improving retrieval quality is therefore a promising direction for future work.

12. Conclusion

This project presents a practical and robust approach to narrative consistency verification in long-form fiction. By combining semantic retrieval with NLI-based verification and conservative aggregation, the system is able to reason over long narratives while avoiding hallucination-prone generation.

The overall design aligns closely with the objectives of Track-A, emphasizing evidence-grounded reasoning, interpretability, and robustness over raw generative capability.