MY REPORT

1. Problem Understanding

In this hackathon, I given very long stories, like full novels. Along with each story, I also given a newly written background story (backstory) for one important character.

My task is very simple to say, but hard to do:

I must decide whether the given backstory matches the events and behavior shown in the novel.

If the backstory fits the story well, I label it "consistent". If it does not fit or breaks the logic of the story, I label it "contradict".

The difficulty comes from the fact that:

- The stories are very long
- Important information is spread across many parts
- Early events affect what can happen later

So, I must check overall logic, not just small sentences.

2. Data Description

The dataset contains two main files:

Training Data

- Each row represents one full story
- It contains:
  - ✓ story_id: unique number for each story
  - ✓ content: the full text of the story
  - ✓ label: whether the backstory is *consistent* or *contradict*

Test Data

- Similar to training data
- But the label column is missing

- My job is to predict the label

The stories are very long, so reading or processing them as a single block is not reliable.

In addition to the training and test CSV files, the dataset also provides full-length novels such as *The Count of Monte Cristo* and *In Search of the Castaways*. These books are used as long-context narrative examples. They help test whether the system can process and reason over very large texts where important information is spread across many parts of the story.

3. Key Idea of My Solution

Instead of reading the full story at once, I follow a human-like approach.

Real-life example:

When I read a long book, I do not decide everything from one page. I read page by page, understand each part, and then decide what the whole story means.

I apply the same idea here.

My main idea:

1. Break each long story into small parts (chunks)
2. Analyze each chunk separately
3. Combine all chunk decisions to make one final story decision

This helps us understand both local details and overall consistency.

4. Text Cleaning

Before using the text, I clean it so the computer can understand it better.

I do the following:

- Convert all text to lowercase
- Remove punctuation marks
- Remove extra spaces
- Make sure all text is treated as plain text

Why this is important:

Just like I clean raw food before cooking, I clean text before using it. Clean text gives more reliable results.

5.Breaking Stories into Chunks

Each story is very long, so I divide it into smaller pieces.

- Each chunk contains a fixed number of characters
- Some overlap is kept so I don’t miss important connections

Why chunking helps:

- Smaller text is easier to analyze
- Important details are not lost
- Logical flow is preserved

Each story becomes a group of chunks.

This chunking strategy is especially useful for very large texts such as full novels. For example, books like *The Count of Monte Cristo* and *In Search of the Castaways* are too long to analyze at once. By breaking them into smaller overlapping chunks, the system can still capture important events and logical constraints across the entire story.

6. Converting Text to Numbers

Computers cannot understand words directly. So, I convert text into numbers using TF-IDF.

Simple explanation:

- Important words get higher value
- Common words get lower value
- Each chunk becomes a list of numbers

This allows the machine learning model to work with the text.

7. Model Training

I use Logistic Regression as my model.

Why this model:

- Simple and reliable
- Easy to understand
- Works well for binary decisions

- Does not overfit easily

The model is trained on chunk-level data, where each chunk has a label based on the story it came from.

8. Chunk-Level Prediction

After training:

- Each chunk is predicted as:
  - ✓ consistent (1)
  - ✓ contradict (0)

This tells us what each small part of the story suggests.

9. Story-Level Decision

A story has many chunks, so I combine chunk predictions.

How I do it:

- For each story, I take the average of all chunk predictions
- If most chunks support consistency → story is consistent
- Otherwise → contradict

Why this works:

This is like voting. If most parts agree, the final answer is reliable.

This step ensures global consistency, which is the main goal of the problem.

10. Test Data Prediction

For the test data, I repeat the same steps:

1. Clean the text
2. Break into chunks
3. Convert text to numbers
4. Predict chunk labels
5. Aggregate to story level

Finally, I create a results.csv file with:

- story_id
- final prediction (consistent or contradict)

This file is used for evaluation.

11. Results

I observed:

- High accuracy at chunk level
- Even better and more stable accuracy at story level

Story-level predictions are more meaningful because the problem is about overall narrative logic, not individual sentences.

12. Limitations

My approach has some limitations:

- Very subtle contradictions spread across many chunks may be missed
- Deep emotional or symbolic meaning is not fully captured

However, the approach is strong, reliable, and well-suited for the given task.

13. Conclusion

In this project, I focused on logical consistency over long narratives.

By breaking stories into chunks and combining predictions carefully, I am able to:

- Handle very long text
- Reduce noise
- Make reliable story-level decisions

My solution is simple, clear, and effective, and it directly addresses the core challenge of the hackathon.