KDSH 2026 Track A Technical Report

Team: Gradient Descenters
Track: Track A - Systems Reasoning with NLP and Generative AI
Submission Date: January 2026

Team Members:

- **Veeky Kumar** (Team Leader) - +917597605761
- **Avinash Kumar Prajapati** - +919928932019
- **Akhilendra Dwivedi** - +919569987852

Executive Summary

This report presents a hybrid RAG-based system for evaluating consistency between character backstories and long-form narratives. The system achieves **63.75% accuracy** on the training set through a combination of Pathway ETL framework, semantic search using sentence transformers, and LLM-based reasoning with Ollama.

Key Results:

- Training Accuracy: 63.75% (51/80 correct predictions)
- Processing Time: ~45 minutes for full dataset (2 novels, 80 examples)
- False Positive Rate: 0% (perfect precision on consistent cases)
- False Negative Rate: 100% (missed all contradictions)

The system demonstrates strong **Track A compliance** through meaningful Pathway integration while maintaining production-ready robustness through graceful fallback mechanisms.

---

1. Problem Understanding

1.1 Task Definition

Given: A complete long-form narrative (novel, 100k+ words) and a hypothetical character backstory (newly written, deliberately plausible).

Determine: Binary classification - Consistent (1) or Contradictory (0) - whether the backstory respects key constraints established in the narrative.

1.2 Core Challenges

Long Context Management: Novels contain 100k+ words, exceeding typical LLM context windows. Relevant evidence may be distributed across multiple chapters, requiring efficient retrieval of pertinent segments.

Causal Reasoning: Must distinguish between correlation and causation, detect logical impossibilities versus plausible additions, and handle ambiguity in underspecified information.

Evidence Aggregation: Integrate signals from multiple text passages, avoid over-reliance on single excerpts, and balance precision with recall.

Narrative Constraints: Track character development over time, respect temporal ordering of events, and detect incompatible motivations or relationships.

---

2. Technical Approach

2.1 System Architecture

The system follows a five-stage pipeline:

Stage 1: Document Ingestion (Pathway ETL) - Read novel files using Pathway's binary format reader, apply chunking transformation (300 words, 50-word overlap), with fallback to native Python if errors occur.

Stage 2: Embedding Generation - Process chunks using all-mpnet-base-v2 (768-dimensional embeddings) in batches of 32 with progress tracking.

Stage 3: Semantic Retrieval - Encode backstory query, compute cosine similarity against all chunk vectors, retrieve top-10 most relevant passages (~3000 words context).

Stage 4: LLM Reasoning - Use qwen2.5:7b via Ollama with temperature 0.1 for consistent reasoning, structured prompt with explicit contradiction criteria, and JSON output parsing.

Stage 5: Output Generation - Format results as CSV with Story ID, Prediction, and Rationale columns.

2.2 Pathway Integration (Track A Requirement)

Implementation Strategy:

```
python
```

```
# Binary read to handle any file encoding
documents = pw.io.fs.read(
    parent_dir,
    glob=filename,
    format="binary",
    mode="static",
    with_metadata=True
)

# Decode and chunk in transformation
def process_file(data, meta):
    text = data.decode('utf-8', errors='ignore')
    return chunk_text_udf(text, meta)

chunks_table = documents.select(
    res=pw.apply(process_file,
                 pw.this.data,
                 pw.this._metadata)
).flatten(pw.this.res)
```

Pathway Benefits Demonstrated: Type-safe column operations, declarative ETL pipeline, easy materialization to CSV for inspection, and production-ready error handling with graceful fallback.

2.3 Semantic Search Strategy

Embedding Model: `all-mpnet-base-v2` with 768 dimensions, trained on sentence-level semantic similarity, processing 17-22 seconds per batch (32 chunks).

Retrieval Logic: Encode query (backstory), compute cosine similarity to all chunks, retrieve top-10 most relevant passages. The top-10 threshold balances context richness (~3000 words) with LLM token limits while staying within the 12k character limit after formatting.

2.4 LLM Reasoning Prompt Design

Core Prompt Structure:

```
Task: Check consistency between Backstory and Novel.

NOVEL EXCERPTS:
[Retrieved top-10 chunks, approximately 3000 words]

BACKSTORY CLAIM:
[User's backstory text]

INSTRUCTIONS:
1. Determine if Backstory CONTRADICTS Novel
2. Contradiction = logical impossibility
   (wrong dates, dead character alive,
   wrong relationship)
3. If backstory adds new plausible details
   then CONSISTENT
4. If novel doesn't mention event
   then CONSISTENT

DECISION:
- Return 0 (Contradict) ONLY if clear
  opposing evidence exists
- Return 1 (Consistent) if plausible
  or not mentioned

JSON OUTPUT:
{"prediction": 0 or 1, "reasoning": "..."}
```

Design Rationale: Explicit criteria reduce ambiguity, conservative default (predict 1 when uncertain) matches observed label distribution, structured JSON output enables reliable parsing, and low temperature (0.1) ensures consistent reasoning.

---

3. Handling Long Context

3.1 Chunking Strategy

Parameters: 300-word chunks with 50-word overlap, resulting in ~2413 chunks for "In Search of Castaways" and ~1857 for "Count of Monte Cristo".

Rationale: 300 words provides sufficient context preservation for complete ideas while maintaining token efficiency within embedding model limits. Fine-grained chunks enable precise matching, while 50-word overlap prevents splitting key phrases.

3.2 Retrieval-Augmented Generation (RAG)

Two-stage process: Coarse retrieval via semantic search identifies top-10 most relevant chunks based on cosine similarity, reducing 4270 total chunks to 10 relevant passages. LLM reasoning then receives only this relevant context, focusing on pertinent evidence while avoiding information overload.

3.3 Memory Management

Batch Processing: Embeddings generated in batches of 32 with real-time progress tracking. Total embedding time is ~40 minutes for both novels, with embeddings cached in memory during session for fast query time (~0.2 seconds per retrieval).

---

4. Distinguishing Causal Signals from Noise

4.1 Signal Detection Strategy

Contradictions Include: Temporal inconsistencies (events in wrong chronological order), state conflicts (character dead in novel but alive in backstory), relationship contradictions (incompatible family ties), and motivation mismatches (actions incompatible with stated beliefs).

Not Contradictions: Underspecification (novel doesn't mention backstory event), plausible additions (new details fitting narrative logic), ambiguous phrasing (multiple valid interpretations), and minor details (spelling variations, approximate dates).

4.2 Evidence Aggregation

The LLM receives 10 chunks from different parts of the novel and must synthesize information across passages, detecting both supporting and contradicting evidence. The balanced prompt explicitly defines contradiction versus consistency, provides decision criteria, and defaults ambiguous cases to consistent.

4.3 Observed System Behavior

Strengths: Zero false positives (never incorrectly marks consistent as contradictory), conservative classification (defaults to "consistent" when uncertain), and perfect performance on clearly consistent cases (51/51 correct).

Weaknesses: Misses all 29 actual contradictions (100% false negative rate), overly conservative tendency accepting plausible-sounding backstories, and insufficient contradiction detection requiring more aggressive prompting or fine-tuning.

---

5. Performance Analysis

5.1 Training Set Results

Confusion Matrix:

	Predicted: 0	**Predicted: 1**
Actual: 0 (Contradict)	0	29
Actual: 1 (Consistent)	0	51

Metrics: Accuracy 63.75% (51/80), undefined precision for class 0 (no positive predictions), 0% recall for contradictions, 100% recall for consistent cases.

Interpretation: System is highly conservative, never risks false positives (good for high-stakes scenarios), but misses all contradictions (needs improvement). Effectively operates as a "predict consistent for everything" classifier.

5.2 Computational Profile

Per-Novel Processing Time:

Task	**In Search of Castaways**	**Count of Monte Cristo**
Pathway Chunking	~1 sec	~1 sec
Embedding Generation	22 min 17 sec	16 min 56 sec
Total Ingestion	**~22.5 min**	**~17 min**

Per-Query Processing: Semantic retrieval (~0.2 sec) + LLM inference (~4 sec) = ~4.2 seconds total per query.

Full Pipeline: Training set (80 examples) in ~45 minutes, test set (60 examples) in ~40 minutes. Primary bottleneck is embedding generation (one-time cost).

5.3 Resource Usage

Memory peaks at ~4GB (embeddings + LLM), disk usage ~500MB (cached embeddings), CPU at 100% during embedding generation, minimal network usage (local Ollama instance).

---

6. Key Design Decisions

6.1 Pathway Integration Approach

Decision: Use binary format with decode-in-transformation.

Rationale: Handles encoding variations robustly, demonstrates Pathway's transformation capabilities, allows graceful fallback without Pathway dependency, and meaningfully meets Track A requirement.

Alternatives Considered: Plaintext format (fragile to encoding issues), token-only import (doesn't demonstrate real usage), custom connectors (over-engineering).

6.2 Embedding Model Selection

Decision: all-mpnet-base-v2.

Rationale: Best quality/speed tradeoff for semantic search, 768 dimensions capture rich semantics, widely used and well-documented, CPU-compatible (no GPU required).

Alternatives Considered: BERT base (lower quality), Sentence-T5 (slower with marginal quality gain), OpenAI embeddings (external API dependency).

6.3 LLM Selection

Decision: qwen2.5:7b via Ollama.

Rationale: Runs locally (no API costs, maintains privacy), 7B parameters balance quality versus speed, strong reasoning capabilities, Ollama provides simple API interface.

Alternatives Considered: GPT-4 (expensive, external dependency), Llama 2 7B (similar performance), Mixtral 8x7B (slower with marginal improvement).

6.4 Conservative Classification Strategy

Decision: Default to "consistent" when uncertain.

Rationale: Training data shows class imbalance (51 consistent versus 29 contradict), false positives may be more costly than false negatives in practical applications, prompt design encourages evidence-based contradictions only.

Observed Impact: Zero false positives (beneficial) but 100% false negatives (problematic), indicating need for rebalancing in next iteration.

---

7. Limitations and Failure Cases

7.1 Known Limitations

Over-Conservative Classification: System predicts "consistent" for everything due to balanced prompt and

conservative temperature, resulting in 0% recall on contradictions. Mitigation requires more aggressive prompt and fine-tuned threshold.

Long Embedding Generation Time: 40 minutes to process both novels due to CPU-only inference and large chunk count impacts cold start time. Mitigation options include GPU acceleration, reduced chunk count, or embedding caching.

Context Window Truncation: Very long backstories (>3000 words) get truncated due to 12k character prompt limit, potentially missing relevant details. Mitigation through hierarchical summarization or dynamic chunking.

Single-Book Retrieval: System only retrieves from specified book, missing potential cross-novel patterns (though not applicable for this specific task).

7.2 Failure Case Analysis

Example: Plausible Contradictions

Backstory claims "Character X was born in Paris in 1820" while novel states "Character X grew up in London from childhood." System incorrectly predicts consistent because LLM interprets these as compatible (born in Paris, moved to London).

Root Cause: Semantic search may not retrieve specific birth location, LLM gives benefit of doubt to plausible explanations, conservative prompt doesn't penalize weak contradictions.

Solution: Add entity extraction step (locations, dates, relationships), implement explicit fact-checking against extracted entities, enable multi-hop reasoning over evidence chains.

7.3 Edge Cases

Ambiguous Phrasing: Novel uses metaphors/symbolism while backstory interprets literally; system may not catch mismatch.

Implicit Contradictions: Backstory claims character is orphan while novel mentions parents indirectly; requires inference over multiple passages.

Underspecified Backstories: Vague backstories ("troubled childhood") are compatible with many interpretations; system correctly predicts consistent.

---

8. Future Improvements

8.1 Short-Term Enhancements (1-2 weeks)

Prompt Re-Engineering: Add explicit contradiction examples, use chain-of-thought prompting, increase temperature slightly (0.1 → 0.3). Expected impact: +15-20% accuracy.

Threshold Tuning: Calibrate confidence scores, adjust decision boundary based on label distribution. Expected impact: +10% accuracy.

Multi-Query Expansion: Generate multiple query variations, retrieve with each variant, aggregate results. Expected impact: Better recall on edge cases.

8.2 Medium-Term Improvements (1-2 months)

Entity-Aware Retrieval: Extract named entities (characters, locations, dates), build knowledge graph of relationships, query graph for fact-checking. Expected impact: +20-25% accuracy.

Hierarchical Summarization: Summarize novel at multiple granularities, use for initial filtering before detailed retrieval. Expected impact: Faster, more accurate retrieval.

Fine-Tuned Classification Head: Fine-tune small model (BERT-base) on task, use as initial filter before LLM reasoning. Expected impact: +30% accuracy, 50% faster.

8.3 Long-Term Research Directions (3-6 months)

BDH Integration (Track B): Explore Baby Dragon Hatchling architecture with persistent state for long-form reasoning and sparse attention over narrative. Expected impact: Novel approach with potential for significant gains.

Causal Graph Construction: Build explicit causal graph of narrative events, query graph for constraint satisfaction. Expected impact: More interpretable, robust reasoning.

Multi-Agent Verification: One agent proposes contradictions while another validates with evidence in adversarial setup. Expected impact: Better calibration, fewer false positives/negatives.

---

9. Reproducibility and Deployment

9.1 Dependencies

Core Requirements: pandas>=2.0.0, pathway>=0.7.0, sentence-transformers>=2.5.0, scikit-learn>=1.3.0, torch>=2.1.0, requests>=2.28.0, tqdm>=4.65.0, numpy>=1.24.0

External Dependencies: Ollama server running locally or via host.docker.internal with qwen2.5:7b model pulled (`ollama pull qwen2.5:7b`)

9.2 Reproducibility Guarantees

Fixed Seeds: sentence-transformers uses deterministic tokenization, PyTorch CPU-only inference is deterministic, Ollama temperature=0.1 ensures consistent reasoning.

Platform Independence: Docker ensures environment consistency, fallback mechanisms handle OS differences, no GPU required (CPU-only operation).

9.3 Deployment Considerations

Production Readiness: Error handling and graceful degradation implemented, progress tracking and logging available, automated pipeline through pipeline.py, but no caching mechanism for embeddings and no API endpoint for real-time inference currently available.

Scalability: Embedding generation is one-time cost, query time scales linearly with chunk count, can parallelize across multiple novels. Bottleneck is LLM inference at 4 seconds per query.

---

10. Conclusion

10.1 Summary of Contributions

This work demonstrates: (1) Track A Compliance through meaningful Pathway integration for ETL pipeline, (2) Robust Engineering with graceful fallbacks ensuring reliability, (3) Effective Semantic Search for long-context handling via retrieval, and (4) Conservative Classification achieving zero false positives with high precision.

10.2 Key Takeaways

Strengths: Robust, production-ready codebase with effective semantic retrieval, fast query time after initial indexing, and clear reproducible pipeline.

Weaknesses: Over-conservative classification with 0% recall on contradictions, long cold-start time of 40 minutes for embedding generation, limited entity-level reasoning, and needs prompt re-engineering for better balance.

10.3 Lessons Learned

Prompt engineering critically impacts outcomes (conservative prompt yields conservative predictions). Semantic search effectively finds relevant context. Pathway adds tangible value through declarative ETL simplification. Class imbalance requires explicit handling strategies.

10.4 Next Steps

Immediate (for competition): Adjust prompt for more aggressive contradiction detection, tune temperature to 0.2-0.3, add explicit contradiction examples in prompt.

Post-competition (for research): Explore entity-aware reasoning, implement causal graph construction, investigate BDH for long-form narrative understanding.

References

1. Pathway Framework: https://pathway.com
2. Sentence-Transformers: https://www.sbert.net
3. Ollama: https://ollama.ai
4. all-mpnet-base-v2: https://huggingface.co/sentence-transformers/all-mpnet-base-v2
5. KDSH 2026 Challenge Documentation

Appendix A: Code Structure

```
KDSH_2026_TrackA/
├── solution.py          # Main system (450 lines)
│   ├── PathwayDocumentStore
│   ├── ConsistencyEvaluator
│   └── main()
├── pipeline.py          # Automation script (80 lines)
├── helpers.py           # Validation utilities (100 lines)
├── requirements.txt     # Dependencies
├── Dockerfile           # Container definition
├── docker-compose.yml   # Orchestration
└── output/              # Generated predictions
```

Appendix B: Sample Predictions

Example 1: True Positive (Correctly Identified as Consistent)

- Book: Count of Monte Cristo
- Backstory: "Edmond Dantès had maritime training before his arrest"
- Prediction: 1 (Consistent)
- Rationale: Novel confirms he was a sailor and ship captain
- Ground Truth: 1 (Correct)

Example 2: False Negative (Missed Contradiction)

- Book: In Search of Castaways

- Backstory: "Captain Grant was rescued from the island by 1862"
- Prediction: 1 (Consistent)
- Rationale: No explicit contradiction found in retrieved context
- Ground Truth: 0 (Incorrect)
- Reason: Novel states rescue happened in 1864

Report Status: Final Submission (Optimized)
Word Count: ~3,800 words (Main Report)
Pages: 10 pages (excluding appendix)
Revision: v2.0 (Condensed and restructured)
Team: Gradient Descenters