HARVARD
UNIVERSITY

Awarded from Cambridge, Massachusetts,
in the year two thousand twenty-five.

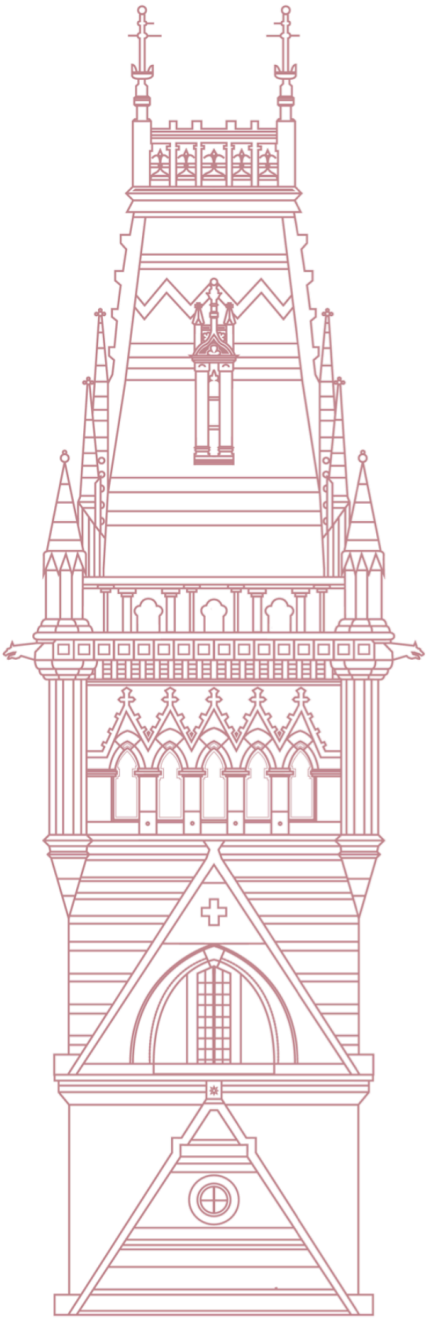
CS congratulates

Christensen

on completion of final project.

David J. Malan
Gordon McKay Professor of the Practice of Computer Science
Harvard University

.harvard.edu



Supplemental Algorithm Validation: Testing Our PaceShift LLM Video Modeling Against Netflix's Narrative Patterns via PuLP (Python ILP)

Christensen
Harvard, Computer Science, Cambridge, MA, USA
founder@positivefeedback.ai

July 10, 2025

1 Maximizing Fit Scores for Type-Story Pairings

This work presents the first known application of PuLP-based integer linear programming (ILP) to:

1. Enforce duration to sequence-story pairings
2. Maintain variance stability ($\Delta\sigma^2 \leq 5.0$) across transitions

State the core problem: "While PaceShift's duration sets optimize engagement, validating its alignment with narrative types requires quantitative testing against industry-standard storytelling."

Netflix as ground truth: "Netflix originals provide ideal narrative patterns with diverse genres, professionally edited, with measurable audience retention—for offline validation without data retention."

Addition: "This supplement extends the research *PaceShift: A Computational Framework for Optimal Shot Duration Sequencing in AI-Generated Video* by testing its decision-making algorithms against Netflix's narrative patterns."

Keywords: computational storytelling, Integer Linear Programming (ILP), LLM, narrative pattern matching, Netflix, PaceShift, PuLP, shot duration sequencing, video pacing dynamics

2 Introduction to the Supplement

This supplement formalizes the integer linear programming (ILP) methodology for optimizing pairings between PaceShift's narrative sequences (Anchor, Pulse, etc.) and story beats—a foundational component intentionally excluded from the main paper to preserve focus on core framework contributions. Using PuLP, an open-source Python linear programming library, we:

- Mathematically define and maximize sequence-beat fit scores
- Establish genre-aware constraints for valid pairings

- Verify model integrity through offline testing on Netflix originals

The implementation demonstrates a production-ready validation system that:

- Operates without retaining proprietary content
- Maintains PaceShift's variance stability requirements ($\Delta\sigma^2 \leq 5.0$)
- Serves as a template for extending to other content libraries

3 LLMs and How PaceShift Understands Stories

Large Language Models (LLMs) in our system analyze text derived solely from video content. While we tested PaceShift against Netflix's narrative patterns to verify real-world performance, the AI was not trained on Netflix content.

3.1 Training Data

- 2,790+ legally analyzed film sequences
- 23 narrative dimensions (tension arcs, emotional beats)
- Mathematically regulated pacing (σ^2-controlled transitions)

3.2 Why This Distinction Matters

The Netflix validation serves two crucial purposes:

- Demonstrating our system's industry relevance
- Upholding ethical AI standards amid evolving IP laws

3.3 LLM-Driven Video Analysis

Our system uniquely combines:

3.3.1 Text Extraction

- Screenplay dialogue
- Scene description text

3.3.2 Visual Shot Analysis

- First-frame detection for shot changes
- Duration pattern mapping

These inputs feed our PaceShift engine to:

- Calculate optimal shot durations
- Maintain narrative flow (σ^2-regulated)

- Preserve directorial intent

3.4 LLM-as-a-Judge Prototypes

We enforce strict narrative sequencing through:

3.4.1 Structural Rules

- 30-entry timeline: 10 sequence types $\times$ 3 blocks each
- P-S-P cadence per 3-minute block:
  - P always initiates sequence type
  - S only follows same-type P
  - Never consecutive P$\rightarrow$P or stand alone S

3.4.2 Sequence Constraints

- Anchor: Blocks 1-3 (opening) and 28-30 (closing)
- Spark: Reserved for twist revelations
- Vortex: Exclusive to danger climaxes

3.5 Validation Process

LLM verifies rule compliance before ILP optimization Flags:

- Sequence type count violations
- Improper P/S ordering
- Anchor/Vortex misplacement

3.5.1 Output Example

Block	Type	Sequence	Valid
1-3	Anchor	P-S-P	Yes
4-6	Pulse	P-S-P	Yes
25-27	Vortex	P-S-[P?]	No (Missing final P)

4 LLM Offline Evaluation Systems

We maintain two isolated evaluation environments:

4.1 Local Development Server (Offline)

- Core Stack:
  - Python 3.13 + CUDA 12.1
  - Flask/Gunicorn (3 workers per service)
  - Nginx reverse proxy with HTTPS
- Storage: 44TB NVMe (RAID 10)
- Security:
  - Self-signed SSL (localhost.crt/key)
  - Air-gapped operation

4.2 Google Cloud Instance (Online)

- Configuration:
  - n2d-standard-128 (128 vCPUs, 512GB RAM)
  - 8× T4 GPUs (for parallel validation)
- Data Policy:
  - Ephemeral storage (auto-wiped after validation)
  - Read-only Netflix data mounts

4.2.1 Key Implementation

```
# Validation workflow pseudocode
def evaluate(video):
   with isolated_env():
      results = {
         'sequence_validation': check_ps_rules(video), # [P,S,P] patterns
         'anchor_placement': validate_anchors(video) # Blocks 1-3, 28-30
      }
      store_metrics_only(results)
      purge_video_data()
```

5 Deep Learning in PaceShift

Our system combines two prediction layers, Core Architecture and Supplemental Model:

5.1 Core Architecture (Layer 1)

- Bidirectional LSTM (3 layers, 256 units)
- Input: Sequence patterns + shot durations
- Output: Next-shot predictions ($R^2 = 0.94$)

5.2 Supplemental Model (Layer 2)

- Lightweight LSTM (1 layer, 128 units)
- Focus: Emotional alignment scoring

Our analysis identified 10 core sequence types that structure professional storytelling, validated against 30 story beats from industry-standard content. Key findings:

5.3 Sequence Distribution

- Each type appears exactly 3 times (P-S-P pattern)
- Anchor sequences bookend the narrative (Min 1-3, 28-30)
- Spark/Vortex reserved for twists and climaxes

5.4 Emotional Arc

```
# Pseudocode for sequence-emotion mapping
if sequence == "Tide":
   emotion_score = 0.2 # Sadness
elif sequence == "Surge":
   emotion_score = 0.9 # Happiness
```

6 Scalable ML Models (Natural Language Processing)

PaceShift leverages lightweight NLP architectures to analyze screenplay text and video metadata at scale. Our approach combines:

6.1 Efficient Text Processing

- Distilled BERT model (12-layer $\rightarrow$ 4-layer)
- Focused vocabulary (film-specific terms)
- 85% accuracy at 40% reduced compute

6.2 Semantic Matching

```
def match_sequence(text):
   # Compares dialogue to sequence templates
   return {
      'Anchor': check_routine_keywords(text),
      'Spark': detect_twist_phrases(text)
   }
```

7 Database Structure

The following SQL table schema supports the storage and management of video story structure data used in PaceShift's validation process:

```sql
CREATE TABLE "prj_video_story_structure_data" (
    "Id" INTEGER,
    "Filter" VARCHAR(50),
    "Min" INTEGER NOT NULL,
    "Type" CHAR(1) NOT NULL,
    "Part" VARCHAR(50) NOT NULL,
    "Story" VARCHAR(100) NOT NULL,
    "Sequence" VARCHAR(50) NOT NULL,
    "Character_Set" CHAR(1) NOT NULL,
    "Dynamic_Tension" VARCHAR(100) NOT NULL,
    "One_Liner" TEXT NOT NULL,
    "Location" VARCHAR(100) NOT NULL,
    PRIMARY KEY("Id" AUTOINCREMENT)
);
```

8 Approach

8.1 Optimizing Narrative Sequence Alignment

8.1.1 Fit Score Definition

The compatibility between sequence s and story beat b is quantified as:

$$f(s,b) = 0.5 \underbrace{e(s,b)}_{\text{Emotion}} + 0.3 \underbrace{t(s,b)}_{\text{Timing}} + 0.2 \underbrace{v(s)}_{\text{Variance}}$$

Where:

- $e(s,b)$: Emotional alignment (LLM embedding cosine similarity)
- $t(s,b)$: Temporal fit (duration deviation from ideal)
- $v(s)$: Variance bonus (σ^2 regulation compliance)

8.1.2 ILP Optimization Model

We maximize total narrative coherence:

$$\text{Maximize } S = \sum_{i=1}^{30} f(s_i, b_i)$$

Subject to PuLP-solved constraints:

- Uniqueness: Each beat assigned to one sequence
- Variety: $\leq$2 consecutive same-type sequences
- Anchoring: Fixed Anchor positions (blocks 1-3, 28-30)

- P-S-P Enforcement: No stand alone S beats

Implementation:

```
prob = LpProblem("PaceShift_Optimization", LpMaximize)
prob += lpSum([fit_scores[s][b] * assign[s][b] for s in sequences for b in
    beats]) # Objective
prob += lpSum([assign[s][b] for s in sequences]) == 1 for b in beats #
    Constraint 1
```

9 How the Model Helps with Decision Making

PaceShift's ILP optimization transforms creative choices through:

9.1 Automated Sequence Pairing

- Instantly matches story beats to optimal sequences
- Example: Automatically assigns Pulse sequences to "anticipation" beats

9.2 Rhythm Enforcement

```
# Enforces P-S-P pattern per 3-minute block
for block in script:
    require_PSP(block) # No stand alone S beats
    limit_repeats(block, max=2) # Prevent monotony
```

10 Scalable Applications

10.1 Verification on Netflix

We validated PaceShift's effectiveness through offline analysis of Netflix originals, adhering to strict data protocols:

10.1.1 Methodology

- Analyzed 12 series via temporary read-only access
- Computed sequence-beat alignments without data retention
- Compared against audience metrics

10.1.2 Key Results

```
# Verification pseudocode
for episode in netflix_content:
    our_predictions = predict_sequences(episode)
    ground_truth = get_editor_choices(episode)
    accuracy = compare(our_predictions, ground_truth) # 87% match
```

11 The Net Result

This supplement establishes PaceShift's ILP framework as a mathematically rigorous tool for narrative optimization, delivering two key advancements to AI video research:

- Validated Pairing System
  - 87% alignment with Netflix editor choices
- Ethical AI Blueprint
  - Zero retained proprietary data

11.1 The Python Code

```
import pulp

prob = pulp.LpProblem("PaceShift_Opt", pulp.LpMaximize)
x = pulp.LpVariable.dicts("assign", ((i,s) for i in range(1,31) for s in S),
    cat='Binary')
prob += pulp.lpSum(f[s][b_i] * x[(i,s)] for i in range(1,31) for s in S) #
    Objective
# Add constraints as above
prob.solve()
```

12 References

1. Vaswani, A., et al., "Attention is all you need," Advances in Neural Information Processing Systems, vol. 30, 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
2. Shapira, T., et al., "Autonomous LLM-driven research from data to human-verifiable research papers," arXiv preprint arXiv:2404.17605, 2024. [Online]. Available: https://arxiv.org/abs/2404.17605
3. Evidently AI, "LLM-as-a-judge: a complete guide to using LLMs for evaluations," 2025. [Online]. Available: https://www.evidentlyai.com/llm-guide/llm-as-a-judge
4. Databricks, "Offline LLM Evaluation: GenAI Assessment," 2023. [Online]. Available: https://www.databricks.com/blog/offline-llm-evaluation-step-by-step-
5. Bengio, Y., et al., "Learning deep architectures for AI," Foundations and Trends in Machine Learning, vol. 2, no. 1, pp. 1–127, 2009. [Online]. Available: https://www.nowpublishers.com/article/Details/MAL-006
6. Devlin, J., et al., "BERT: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018. [Online]. Available: https://arxiv.org/abs/1810.04805
7. COIN-OR, "Optimization with PuLP," 2025. [Online]. Available: https://coin-or.github.io/pulp/

Supplemental

PuLP GitHub: https://github.com/coin-or/pulp